4-2-2018

Image Processing Applications in Real Life: 2D Fragmented Image and Document Reassembly and Frequency Division Multiplexed Imaging

Houman Kamran Habibkhani
Louisiana State University and Agricultural and Mechanical College, hkamra1@lsu.edu

IMAGE PROCESSING APPLICATIONS IN REAL LIFE: 2D FRAGMENTED IMAGE AND DOCUMENT REASSEMBLY AND FREQUENCY DIVISION MULTIPLEXED IMAGING

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The School of Electrical Engineering and Computer Science
Division of Electrical and Computer Engineering

by
Houman Kamran Habibkhani
B.S., Sharif University of Technology, 2007
M.S., Southern University and A&M College, 2009
May 2018

ACKNOWLEDGMENTS

I would like to thank Louisiana State University for giving me the opportunity and providing me with the environment to continue my higher education.

I would also like to thank Dr. Li as my main advisor for guiding me through the project of 2D fragment image reassembly with his wealth of knowledge and experience.

Special thanks go to Dr. Gunturk for his guidance and wisdom through the project of frequency division multiplex imaging.

I would also like to take this opportunity to thank Dr. Wei, Dr. Sun, Dr. Zhang and Dr. Launey for doing me the honor of being a part of my defense committee.

TABLE OF CONTENTS

ABSTRACT

In this era of modern technology, image processing is one the most studied disciplines of signal processing and its applications can be found in every aspect of our daily life. In this work three main applications for image processing has been studied.

In chapter 1, frequency division multiplexed imaging (FDMI), a novel idea in the field of computational photography, has been introduced. Using FDMI, multiple images are captured simultaneously in a single shot and can later be extracted from the multiplexed image. This is achieved by spatially modulating the images so that they are placed at different locations in the Fourier domain. Finally, a Texas Instruments digital micromirror device (DMD) based implementation of FDMI is presented and results are shown.

Chapter 2 discusses the problem of image reassembly which is to restore an image back to its original form from its pieces after it has been fragmented due to different destructive reasons. We propose an efficient algorithm for 2D image fragment reassembly problem based on solving a variation of Longest Common Subsequence (LCS) problem. Our processing pipeline has three steps. First, the boundary of each fragment is extracted automatically; second, a novel boundary matching is performed by solving LCS to identify the best possible adjacency relationship among image fragment pairs; finally, a multi-piece global alignment is used to filter out incorrect pairwise matches and compose the final image. We perform experiments on complicated image fragment datasets and compare our results with existing methods to show the improved efficiency and robustness of our method.

The problem of reassembling a hand-torn or machine-shredded document back to its original form is another useful version of the image reassembly problem. Reassembling

a shredded document is different from reassembling an ordinary image because the geometric shape of fragments do not carry a lot of valuable information if the document has been machine-shredded rather than hand-torn. On the other hand, matching words and context can be used as an additional tool to help improve the task of reassembly. In the final chapter, document reassembly problem has been addressed through solving a graph optimization problem.

CHAPTER 1.
FREQUENCY DIVISION MULTIPLEXED IMAGING

Frequency division multiplexed imaging (FDMI) [20] enables capturing multiple or sub-exposure images through optical modulation and Fourier domain space allocation. Through optical modulation, different band-limited images are placed at non-overlapping frequency sub-bands, allowing simultaneous capture of multiple images. These images can then be recovered through band-pass filtering provided that there is no aliasing. In [20] an optical setup to capture multiple images is presented. Here, another optical setup, specifically to capture sub-exposure images, is presented. This is achieved through pixel-by-pixel exposure control, implemented by using a Texas Instruments (TI) DMD chip.

First, the FDMI idea is briefly explained and then the TI DMD based implementation is discussed in detail.

1.1 Introduction

In a variety of image processing and computer vision applications, it is necessary to capture multiple images of a scene. For example, to form a color image, at least three spectral channels have to be captured. This is done by using either multiple sensors or a color filter array that allocates one color component for each pixel. To form a video, on the other hand, multiple images are sequentially captured by a sensor. These two applications exemplify the two main approaches of capturing multiple images: space allocation and time allocation. In the color image capture application, we do space allocation: there is a color filter array in front of the sensor, assigning a single color component to a pixel; or there are multiple sensors, each assigned to a color channel. In the video capture application, we do time allocation: a time slot is allocated to each video frame. In this work, we discuss a third approach: frequency division multiplexed imaging.

Inspired from the frequency division multiplexing technique in communication systems, authors in [20] propose to optically modulate and place different band-limited images at non-overlapping frequency sub-bands, allowing simultaneous capture of multiple images. These images can then be recovered through band-pass filtering. The underlying idea of FDMI is illustrated in Figure 1.1. Suppose that we have two band-limited images: $\mathrm{I}_1(\mathrm{x},\mathrm{y})$ and $\mathrm{I}_2(\mathrm{x},\mathrm{y})$. We modulate these images with masks $\mathrm{H}_1(\mathrm{x},\mathrm{y})$ and $\mathrm{H}_2(\mathrm{x},\mathrm{y})$, and capture the sum of the modulated images:

$$\mathrm{I}(\mathrm{x},\mathrm{y}) = \mathrm{H}_1(\mathrm{x},\mathrm{y})\mathrm{I}_1(\mathrm{x},\mathrm{y}) + \mathrm{H}_2(\mathrm{x},\mathrm{y})\mathrm{I}_2(\mathrm{x},\mathrm{y}) \tag{1.1}$$

where the masks are sinusoids in horizontal and vertical directions: $\mathrm{H}_1(\mathrm{x},\mathrm{y}) = \mathrm{a} + \mathrm{b}\cos(2\pi\mathrm{u}_0\mathrm{x})$ and $\mathrm{H}_2(\mathrm{x},\mathrm{y}) = \mathrm{a} + \mathrm{b}\cos(2\pi\mathrm{v}_0\mathrm{y})$, where a and b are positive constants with the condition $\mathrm{a} \geq \mathrm{b}$ so that the masks are non-negative and therefore can be realized optically, and u_0 and v_0 are the spatial frequencies of the masks. The Fourier transforms of the masks are $\widehat{\mathrm{H}}_1(\mathrm{u},\mathrm{v}) = \mathrm{a}\delta(\mathrm{u},\mathrm{v}) + \left(\frac{\mathrm{b}}{2}\right)(\delta(\mathrm{u}-\mathrm{u}_0,\mathrm{v}) + \delta(\mathrm{u}+\mathrm{u}_0,\mathrm{v}))$ and $\widehat{\mathrm{H}}_2(\mathrm{u},\mathrm{v}) = \mathrm{a}\delta(\mathrm{u},\mathrm{v}) + \left(\frac{\mathrm{b}}{2}\right)(\delta(\mathrm{u},\mathrm{v}-\mathrm{v}_0) + \delta(\mathrm{u},\mathrm{v}+\mathrm{v}_0))$. Defining $\hat{\mathrm{I}}_1(\mathrm{u},\mathrm{v})$ and $\hat{\mathrm{I}}_2(\mathrm{u},\mathrm{v})$ as the Fourier transforms of $\mathrm{I}_1(\mathrm{x},\mathrm{y})$ and $\mathrm{I}_2(\mathrm{x},\mathrm{y})$, respectively, the Fourier transform of the captured image $\mathrm{I}(\mathrm{x},\mathrm{y})$ is

$$\hat{\mathrm{I}}(\mathrm{u},\mathrm{v}) = \mathrm{a}\left[\hat{\mathrm{I}}_1(\mathrm{u},\mathrm{v}) + \hat{\mathrm{I}}_2(\mathrm{u},\mathrm{v})\right] + \left(\frac{\mathrm{b}}{2}\right)\left[\hat{\mathrm{I}}_1(\mathrm{u}-\mathrm{u}_0,\mathrm{v}) + \hat{\mathrm{I}}_2(\mathrm{u}+\mathrm{u}_0,\mathrm{v})\right] + \left(\frac{\mathrm{b}}{2}\right)[\hat{\mathrm{I}}_1(\mathrm{u},\mathrm{v}-\mathrm{v}_0) + \hat{\mathrm{I}}_2(\mathrm{u},\mathrm{v}+\mathrm{v}_0)]. \tag{1.2}$$

Provided that the images $\mathrm{I}_1(\mathrm{x},\mathrm{y})$ and $\mathrm{I}_2(\mathrm{x},\mathrm{y})$ are band-limited and $(\mathrm{u}_0,\mathrm{v}_0)$ are large enough such that there is no overlap of individual terms in $\mathrm{I}(\mathrm{u},\mathrm{v})$, we can recover the images through band-pass filtering the Fourier transform of the capture image $\mathrm{I}(\mathrm{x},\mathrm{y})$.

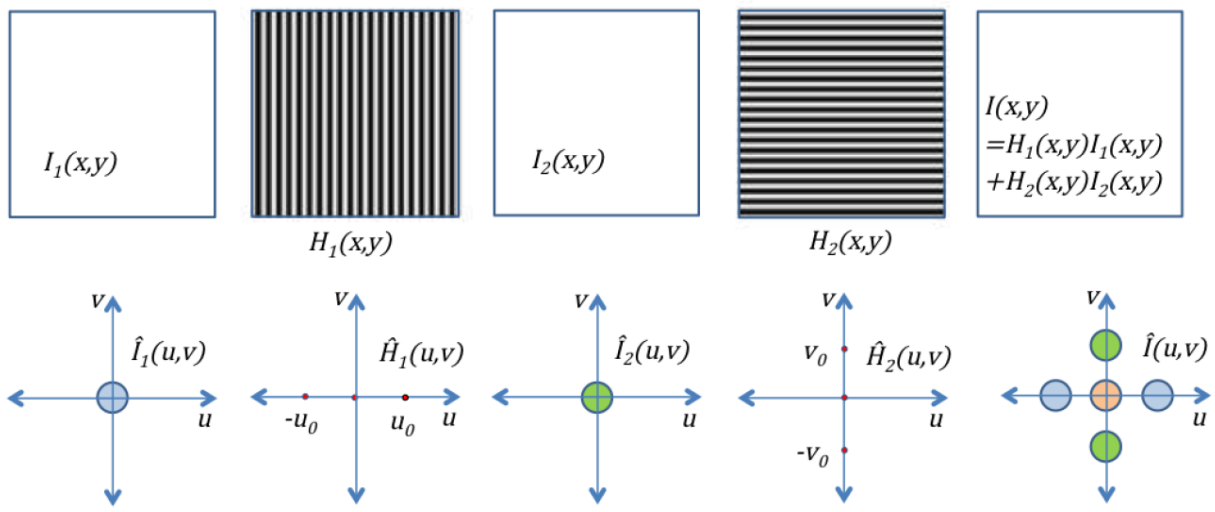


Figure 1.1: Illustration of frequency division multiplexed imaging (FDMI) idea. Assuming that the image are band-limited, whose Fourier transforms are indicated by blue and green circles, and the spatial frequencies (u_0, v_0) of the masks are large enough, the modulated image has Fourier transform $\hat{I}(u,v)$ where the individual Fourier components can be recovered.

1.2 Related Work

In recent years, extensive research has been conducted in the field of computational photography. In general, computational photography is the field of capturing multiple images of the same scene for the purpose of creating a better image of that scene by extracting desirable parts from the taken images and combine them into the final result image.

Debevec and Malik in [64] proposed a novel idea to from an image in which a wide range of radiance map is present. To accomplish such a task, a sequence of images with varying exposure times are captured. The Image with large exposure period will capture the information from the dark part of the scene accurately but will saturate in the bright parts. On the other hand, the image with small exposure period, will capture the bright part of the scene and will lose information in the dark part. By conscious combination of all these images, a final image is synthesized that will have the information from the dark part of

the scene from images with large exposure time and the information from the bright part of the scene from images with small exposure time. Battiato et.at. in [75] and Bandoh et.al. in [76] discuss the more recent advanced made in this topic.

Another application of computational photography can be found in the subject of focus stacking. Many cameras provide insufficient control over depth of field. Some have a fixed aperture; others have a variable aperture that is either too small or too large to produce the desired amount of blur. To overcome this limitation, one can capture a focal stack, which is a collection of images each focused at a different depth, then combine these slices to form a single composite that exhibits the desired depth of field. Jacobs et.al. in [77] propose a theory of focal stack compositing to computer images with extended depth of field. Coded aperture imaging [78], panoramic imaging [79], and stereoscopic imaging are other well-known applications in computational photography.

1.3 FDMI Applications in the Form of Experiments

To explain the FDMI idea more clearly, three different experiments has been designed in [20] that will help us understand different applications of FDMI idea. In the first experiment, the light modulators are static and will not change during the exposure time. Using this type of setup and with the help of a beam splitter, multiple images are captured in a single shot. In the second experiment, light modulators are dynamic, meaning that they change during the exposure time of a single shot. Using this type of optical setup, sub-exposure images, as well as full-exposure images, are captured in a single shot and then studied. Finally, in the third experiment, using dynamic light modulators as in experiment two, different parts of the scene's dynamic range are captured for the purpose of high dynamic range (HDR) imaging.

1.3.1 Experiment 1: Capturing Multiple Images with Static Light Modulators

Figure 1.2 depicts the optical setup that makes the capture of multiple images with a single shot possible. As is shown, using a beam splitter the camera is focused on two intermediate image planes simultaneously. One part of the scene is formed on the first intermediate image plane and the other part of the scene is formed on the second intermediate image plane. A horizontal grating is placed to the first image plane and a vertical grating is placed on the second image plane. As the images are formed on these image planes, they are modulated with these horizontal and vertical gratings respectively and, as a result, the camera captures the sum of these modulated images.

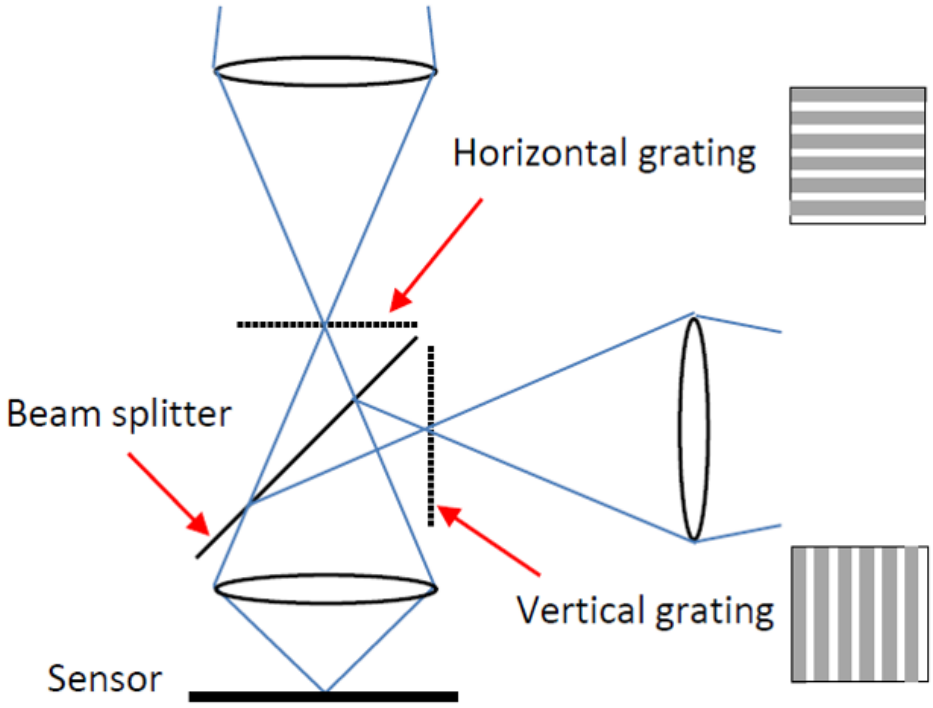


Figure 1.2: Two images are captured simultaneously using static light modulators.

The results of this experiment are presented in figure 1.3. Figure 1.3a is showing the beam splitter and the static horizontal and vertical gratings. Based in figure 1.2, the

images are supposed to form on the intermediate focus planes where the gratings are located. To mimic image formation on the focus planes, two images, shown in figure 1.3b, are chosen as the two scenes to be captured simultaneously and are places right after the gratings instead of using a focusing front lens as illustrated in figure 1.2. Figure 1.3c shows the actual image captured through the beam splitter and figure 1.3d shows it Fourier transform. By applying band-pass filters as shown in figure 1.3d, the first and second scenes can be recovered separately as illustrated in figure 1.3e and figure 1.3f.

As FDMI is explained previously, the gratings are assumed to be sinusoids. But the gratings used in the this experiments are square waves. The Fourier transform of a square wave with frequency u_0 is not two impulses anymore but is an impulse train with decaying magnitudes, $\sum_k \mathrm{sinc}\left(\frac{u}{u_0}\right)\delta(u - ku_0)$. This is also visible in the zoomed-in part of figure 1.3d. To maximize the signal to noise ratio, the location of the first impulse is picked for band-pass filtering.

Looking at figure 1.3d, we notice that the entire frequency range is not utilized; if we had used spatial gratings with higher spatial frequency (lines per inch), we would have achieved a better separation between different image components. The highest possible grating frequency is such that the grating on the image plane has a period of two pixels, which corresponds to $u_0 = 1/2$ on the image plane.

Input images need to be band-limited in order to avoid any aliasing during the multiplexing process. To make sure that that is the case, a low-pass filter needs to be applied to input images. One way of achieving low-pass filtered input images in our specific setup is not to bring the target images to perfect focus in the intermediate focus planes where the gratings are located and where the images where supposed to form at the first place.

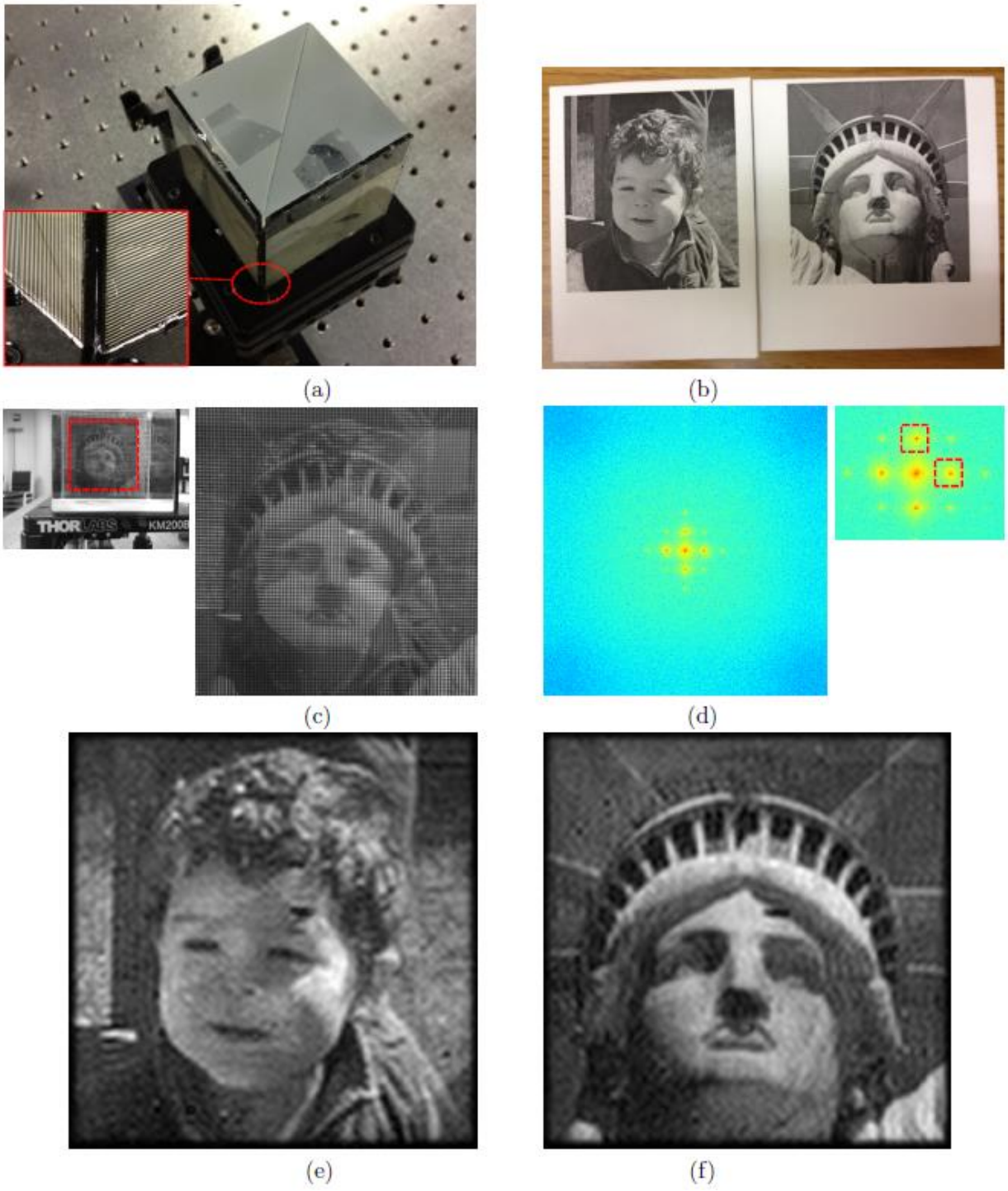


Figure 1.3: Demonstration of the frequency division multiplexed imaging idea. Out-of-focus blur, will reduce the high-frequency components of target images as it acts as a low-pass filter, and, as a result, aliasing artifacts are kept under control.

1.3.2 Experiment 2: Capturing Sub-Exposure Images with Dynamic Light Modulators

Figure 1.4 depicts the design for optical setup for another application based on the FDMI idea. In this experiment, a light modulator is placed on the intermediate focus plane as in experiment 1. But the difference is that the light modulator here is dynamic meaning that it will change pattern during the exposure time of a single shot. As the pattern changes, each sub-exposure image is modulated by a different pattern and therefore in placed at a different location in the Fourier domain. These sub-exposure images, then, can be recovered through appropriate band-pass filtering as we did in experiment 1.

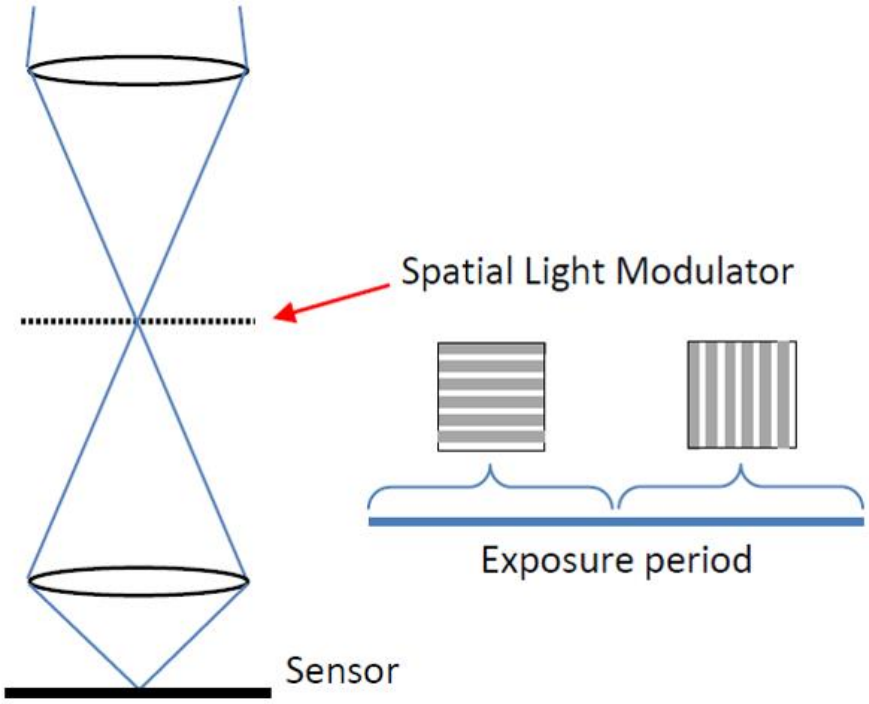


Figure 1.4: Sub-exposure images are captured in a single exposure period

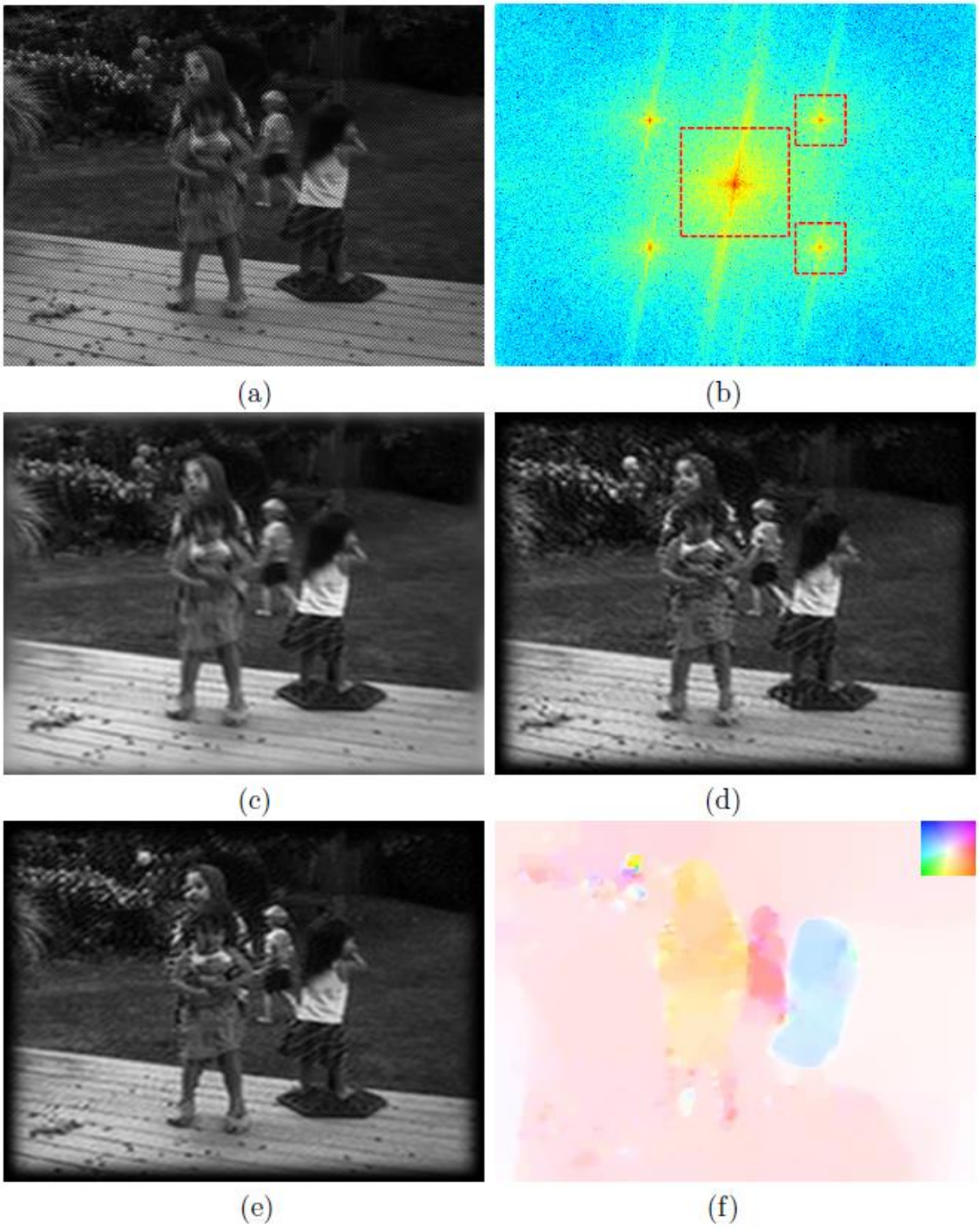


Figure 1.5: Extraction of sub-exposure images.

To demonstrate this idea, simulated data is used for this experiment. Two images of a video sequence is picked to simulate the motion in a scene during the exposure period of

the shot. These images are modulated with orthogonal gratings, one at $+45$ and another at -45 degrees, and are then averaged to simulate the capture process. The result is shown in figure 1.5a. Figure 1.5b is showing the Fourier transform of the captured image. Using a low-pass filter, the image shown is figure 1.5c is recovered. This recovered image is the image that would have been captured assuming there were no gratings to begin with. On the other hand, using band-pass filters images shown in figure 1.5d and figure 1.5e are recovered. The first image corresponds to the first sub-exposure period and the second image corresponds to the second sub-exposure period. These images can be used to different purposes, including scene analysis, optical flow estimation, and motion deblurring. Figure 1.5f is the result of optical flow estimation between the two sub-exposure recovered images. The visible artifacts in the recovered images are due to aliasing and also due the fact that some high frequency components are cut off by the filters.

In addition to testing the idea with simulated data, is needs to be proved with real data as well. To prove the feasibility of the idea, real optical set up is needed. A gray-scale LCD projection panel, connected to a PC with a VGA cable is used. This way, anything that is on the PC display will also be displayed on the panel as well. Therefore, we can form any pattern on the panel as shown in figure 1.6. A camera is focused on the panel; a printout image is placed right after the panel. This imitates an image formed on the focus plane. The pattern formed on the panel is a horizontal grating for the first half of the exposure and vertical grating for the second half. The printout image is slightly moved during the exposure period. The captured image and its Fourier transform are shown in figure 1.7a and figure 1.7b. The motion-blurred image obtained from the center portion of the Fourier

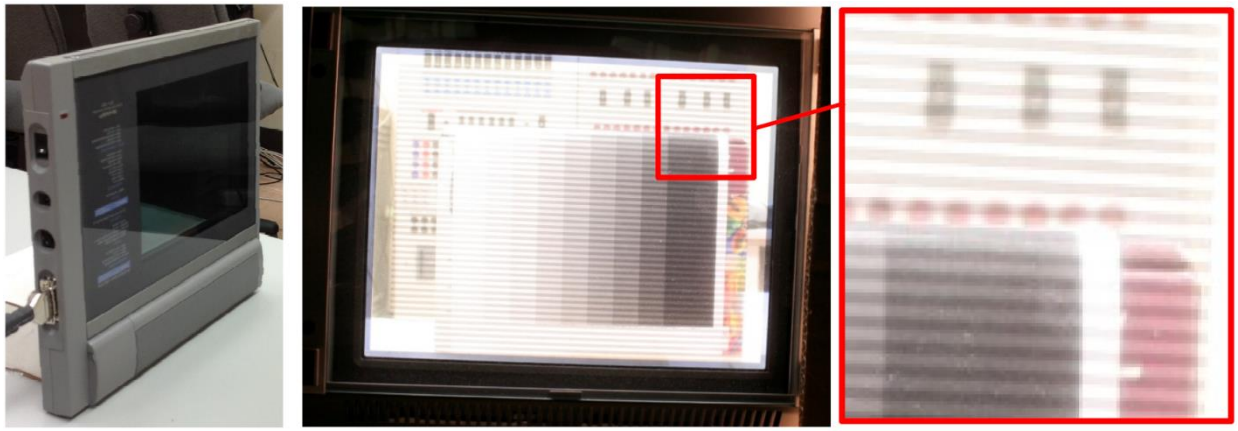


Figure 1.6: LCD projection panel for spatial light modulation. (a) Sharp QA-75 LCD projection panel, (b) camera focuses on the projection panel, which modulates light based on the pattern displayed on the panel.

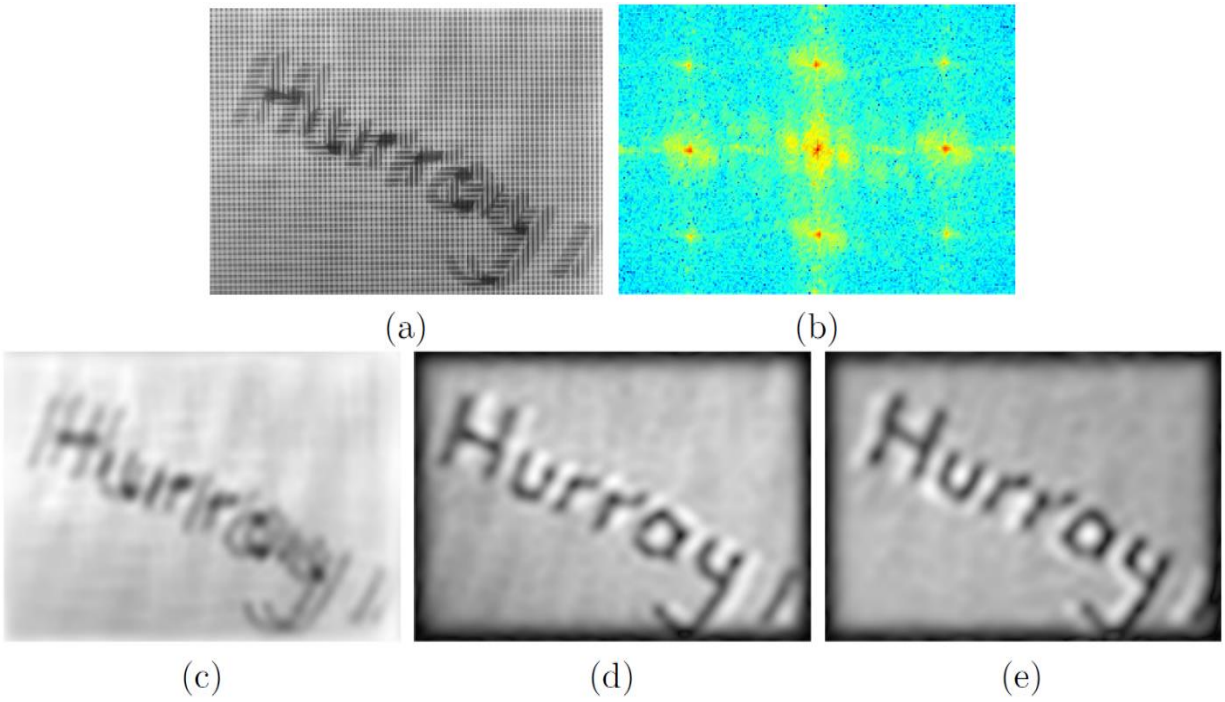


Figure 1.7: Extraction of sub-exposure images.

transform is given in figure 1.7c. The images modulated in the first and second half of the exposure period are shown in figure 1.7d and figure 1.7e.

1.3.3 Experiment 3: Capturing HDR Image with Dynamic light modulator

FDMI idea can be used in creating high dynamic range imaging. While photographing a natural scene, sometimes, we need to increase the exposure period to capture that part of the dynamic range that is dark. Nevertheless, as a result of the long exposure period, the bright part of the dynamic range will end up saturated in the captured image. We propose to modulate the scene image with a horizontal grating for the first half of the exposure period and to have clear screen for the second half of the exposure period. As a result, this sub-exposure modulated image from the first half of the exposure period will contain valid information for the bright part of the dynamic range; and the full exposure image will contain valid information for the dark part of the dynamic range. This gives us the opportunity to create a HDR image using the recovered sub-exposure and full exposure images. This optical setup is displayed in figure 1.8. The region of interest is a printout with bands of different brightness as shown in figure 1.9c which is cropped out of the image in figure 1.9b. Figure 1.9d is the Fourier transform of the cropped image. After a low-pass filter is applied to the central part of the Fourier transform, the recovered image is shown in figure 1.9e; and figure 1.9f is showing the result of applying band-pass filter to the Fourier transform. Comparing figure 1.9e and figure 1.9f, we see that parts of the image that are over-exposed in figure 1.9e due to the full exposure, are well-exposed in figure 1.9f. On the other hand, well-exposed parts in figure 1.9e are under-exposed in figure 1.9f due to shortness of the sub-exposure period. For the next two sections, experiment 3 is repeated, but, this time, with a more accurate real-life setup.

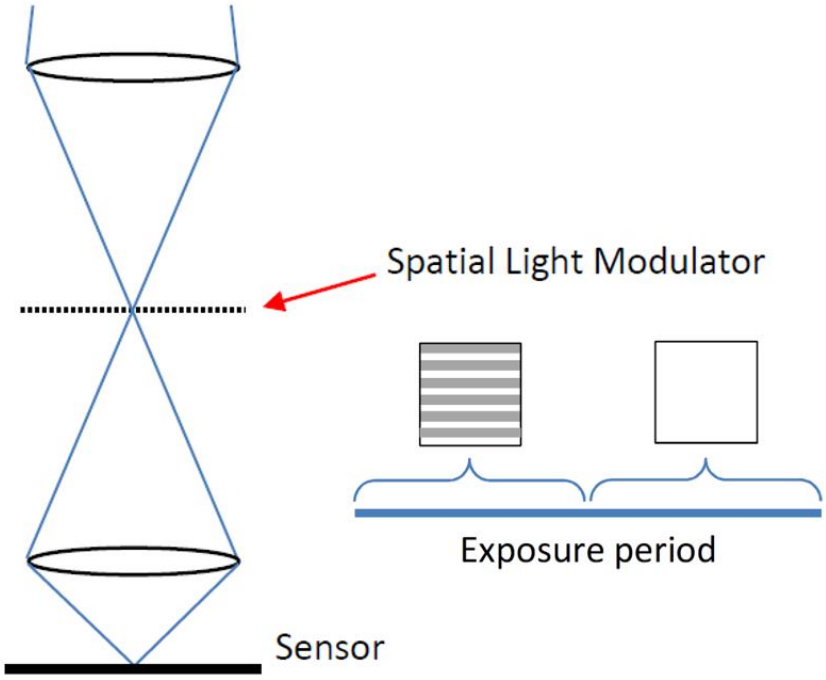


Figure 1.8: In the first half of exposure period, horizontal grating is applied; in the second half, the panel screen is clear.

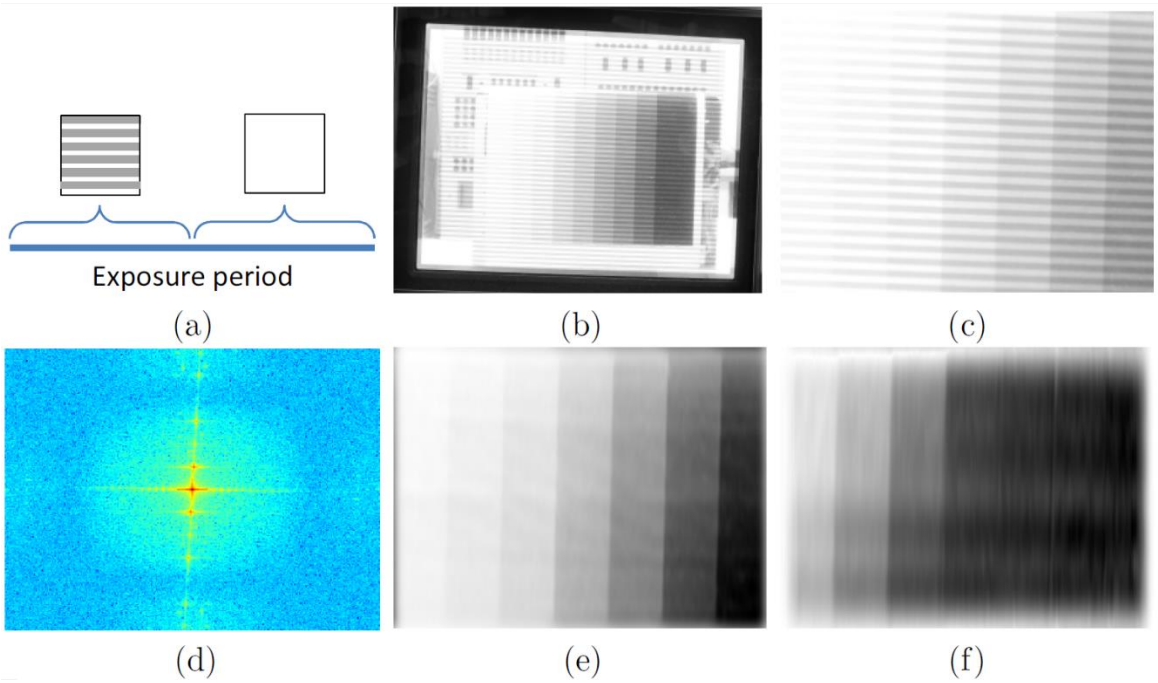


Figure 1.9: Capturing different part of dynamic range.

1.4 Implementing FDMI Using TI DMD

This section presents an optical setup based on the Texas Instruments DLP technology to capture sub-exposure images with the FDMI idea. A Texas Instruments DLP Lightcrafter is used as the optical modulator. TI DLP Lightcrafter is a compact evaluation system for integrating projected light into industrial, medical and scientific applications. Developers can create and display high-speed pattern sequences through Lightcrafter's application program interface (API) and graphical user interface (GUI). The pattern designed by the developer, horizontal or vertical grid in our case, will be formed on the digital micromirror device (DMD) of the Lightcrafter. The goal is to form an image on the DMD while it is bearing the specified grid pattern. The DMD will act as a spatial light modulator and will be able to modulate the amplitude and direction of incoming light. Therefore, the image focused on the DMD will be modulated with the grid pattern that is formed on the DMD.

The DMD on the Lightcrafter is composed of a two-dimensional array of 1-bit CMOS memory cells. These cells are organized in a grid of 608 memory cell columns by 684 memory cell rows. Optically, the DMD consists of 684×608 highly reflective micromirrors, organized in a two-dimensional array. Each individual micromirror is placed on top of a corresponding CMOS memory cell. Each aluminum micromirror is switchable between two discrete angular positions, specifically $-12°$ and $+12°$. The angular positions are measured with respect to $0°$ flat reference when the mirrors are placed in their inactive state, parallel to the array plane. The tilt direction is perpendicular to the hinge-axis.

The angular position of each micromirror is determined based on the binary state of the corresponding CMOS memory cell contents. Writing logic 1 into a CMOS memory cell will

result in the corresponding micromirror to change to the on-state, that is, $+12°$ angular position. On the other hand, writing logic 0 into a CMOS memory cell will cause the corresponding micromirror to switch to the off-state, that is, $-12°$ angular position. We use dynamic spatial light modulation to capture sub-exposure information from an image. Through the GUI that comes with the Lightcrafter, we can specify the pattern on the DMD as a function of time. In the first half of the exposure period, there will be no grating on the DMD, that is, all the incoming light will be reflected. In the second half of the exposure time, there will be vertical grating pattern on the DMD, that is, all light from some pixels will reflected and no light from the others will be reflected. As a result, the second half of the exposure will be modulated and eventually be recovered. If we would like to modulate the first half, then we need to apply another grating pattern, e.g. horizontal grating, in the first half of the exposure period.

Figure 1.10 illustrates the optical setup to capture the sub-exposure information from the scene. An objective lens forms an image of the scene on the DMD. Based on the grating pattern on the DMD, the reflected light will be spatially modulated. A camera, focused on the DMD, captures the modulated image. The beam splitter makes it possible to use the DMD as the image/focal plane for both the object side and sensor side. One is forming an image of the scene on the DMD and the other is capturing the modulated light. Figure 1.11a and figure 1.11b show the actual setup the we have used to conduct the experiments, from two different perspectives.

Figure 1.12a is an image of the DMD when a vertical grid is formed. The image formed on the DMD through the lens is due to white diffuse light. Figure 1.12b shows a close- up region of the DMD. The grating pattern is visible. Note that the gratings used in this

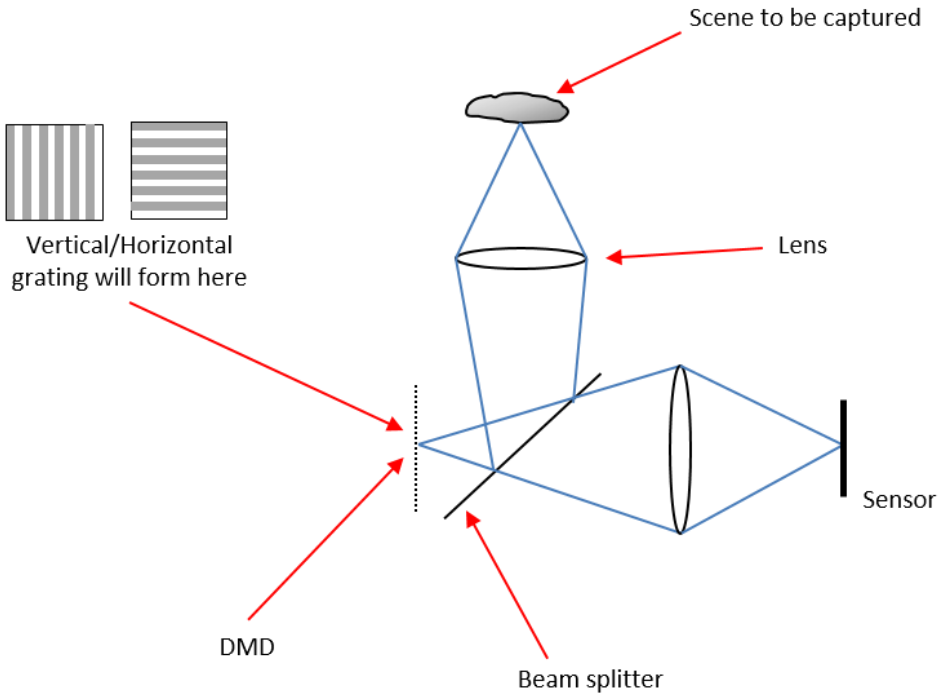


Figure 1.10: Optical configuration for implementing FDMI idea using DMD. Using a beam splitter, both the camera and the imaging lens are focused on the same plane. DMD is used as an intermediate focal plane for both optical components. At the same time, DMD is used as the spatial light modulator. The image of the scene, formed on the DMD through the imaging lens, is modulated by the pattern on the DMD.

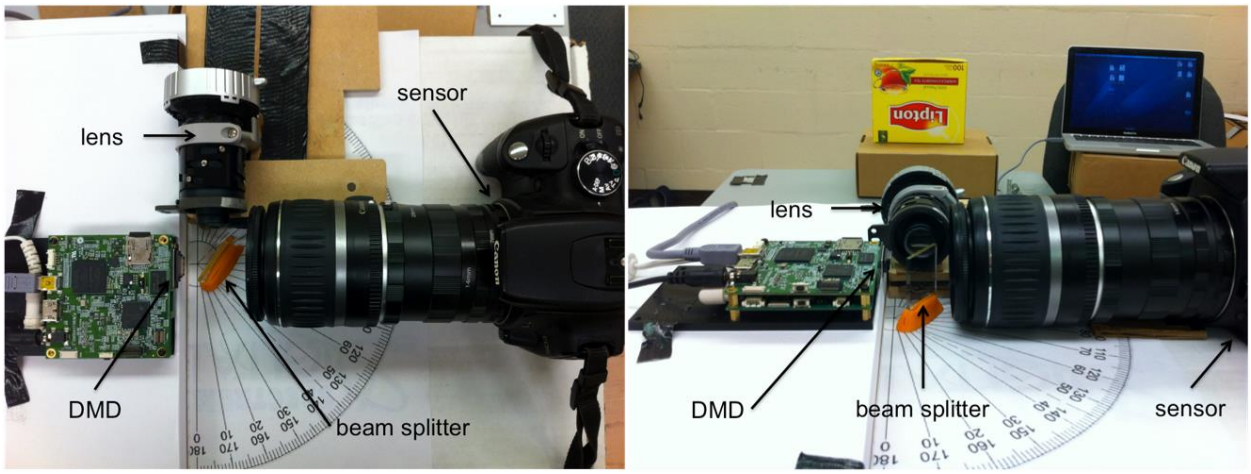


Figure 1.11: The actual setup used to conduct the experiment. (a) Top view, (b) Side view.

experiment are not sinusoids but square waves. The Fourier transform of a square wave with frequency u_0 is an impulse train with decaying magnitudes, $\sum_k \mathrm{sinc}\left(\frac{u}{u_0}\right)\delta(u - ku_0)$. Further zooming in, as shown in figure 1.12c, we recognize that the vertical pattern has a period of eight pixels on the image plane, which corresponds to $u_0 = 1/8$ in the Fourier domain. This results in the impulses on the Fourier transform of the signal to happen at multiples of 1/8. Figure 1.12d illustrates the Fourier transform of the signal.

In order to better utilize the entire frequency range, we should use a grating with spatial frequency as high as possible. We would have achieved the most separation between impulses if we could have created a grating which has a period of two pixels on the image plane. The Fourier transform of such a signal will be an impulse train for which the impulses occur at multiples of $u_0 = 1/2$. This will essentially utilize half of the entire bandwidth. To achieve this best possible scenario, it is critical to use high-precision tools, and register the sensor pixels and DMD pixels and keep a sharp focus for the entire image plane at the same time. Our setup was not implemented with such high-precision tools; therefore, we demonstrate the idea in a level below the best possible.

1.5 Experimental Results

Figure 1.13 shows the results of the experiment. Figure 1.13a displays the image of the scene formed on the DMD through imaging lens. No pattern is created on the DMD when this image is captured; that is, this image is the ground truth signal, against which we can evaluate the quality of our results. Figure 1.13b is the image captured during the actual experiment. In the first half of the exposure period there is no grating; in the second half of the exposure, there is vertical grating. The object is quickly moved from its original position to another position at the beginning of the second half of the exposure.

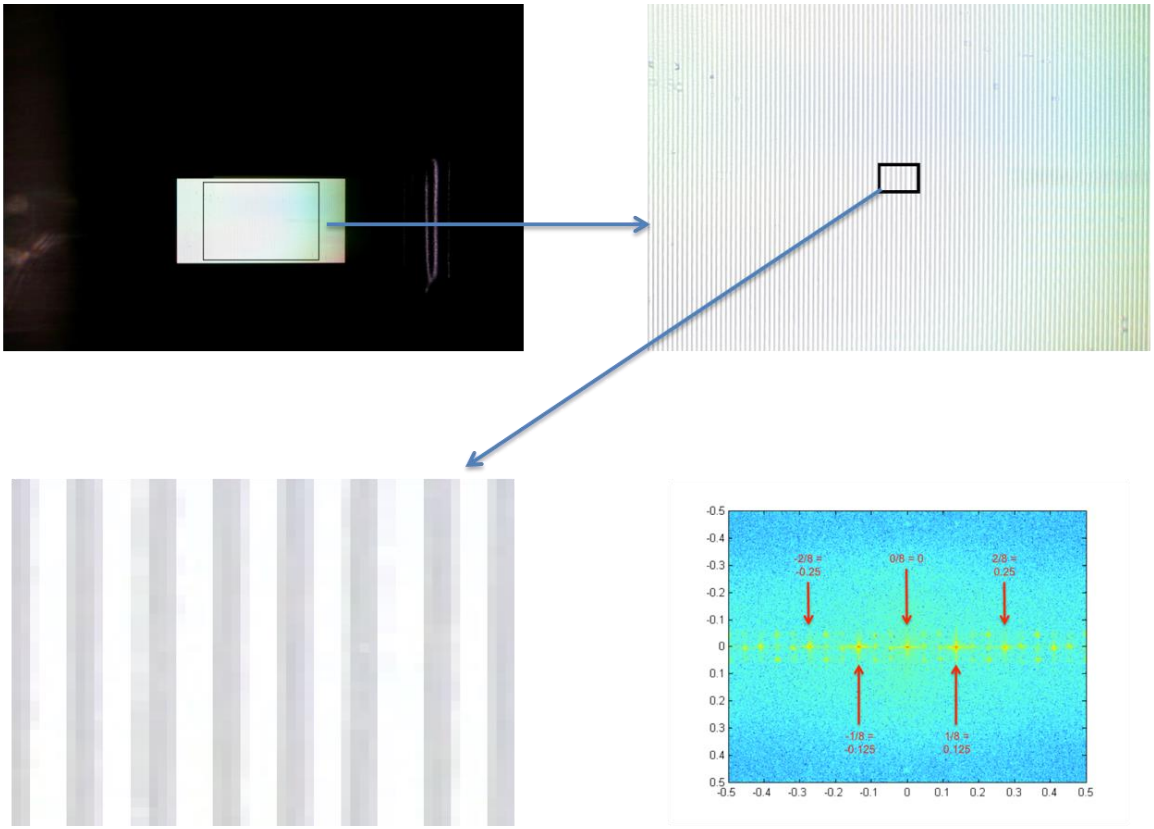


Figure 1.12: Square wave grating pattern formation on the DMD.
(a) The image captured from DMD when a vertical grating pattern is formed on the DMD and the scene is white diffuse light, (b) DMD portion cropped out from the image, (c) A zoomed-in portion of the modulated image, (d) Fourier transform of the modulated image (square wave). Fourier transform of a square wave, with period of 8 pixels on image plane, is an impulse train where the impulses happen at multiples of 1/8.

In figure 1.13c, we have the Fourier transform of the image. We extracted the low-pass and the band-pass components using the filters shown. The low-pass component is essentially the image that would be captured as if there was no spatial light modulation; this component is shown in figure 1.13d. Note that the Fourier transform of the signal we are trying to recover is repeatedly placed at the locations of the impulses of the impulse train due to square wave pattern. We pick the location of the first impulse for band-pass filtering because it has the highest signal to noise ratio. The image extracted from the band-pass component is shown in figure 1.13e. It is recognizable that our ground truth

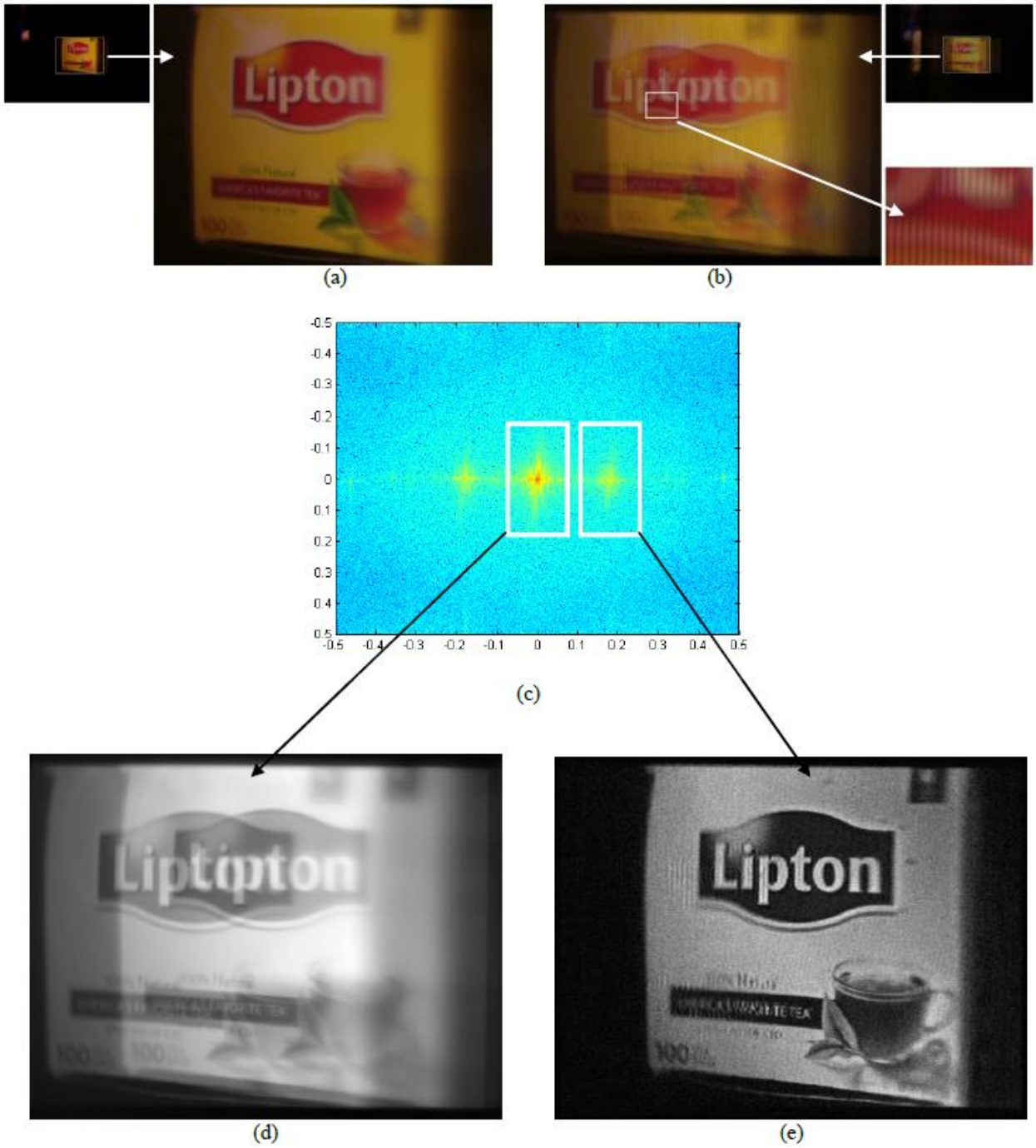


Figure 1.13: Experiment illustrating the FDMI idea with dynamic spatial light modulation. (a) This image is captured without any grating and serves as the ground truth, (b) This image is captured during the actual experiment. During the first half of exposure time there is no pattern on the DMD; and during the second half there is a vertical grating pattern, (c) Fourier transform of the image given in (b), (d) the image recovered from the low-pass component, (e) the image recovered from the band-pass component.

image and this image, which is recovered from a modulated image, have about the same level of detail.

1.5.1 Analysis and Comparison

In chapter 1, we introduced a novel idea, frequency division multiplexed imaging, to create a new imaging system, which will help us in different applications in the field of computational photography.

The experiments in chapter 1 are designed to illustrate the FDMI idea, which is an alternative to space or time allocation in capturing multiple images. The experiments are conducted with fixed glass gratings and LCD-based dynamic gratings. With these optical setups, it is difficult to register the sensor pixels and gratings, and achieve the best possible results. The results can be improved by using more accurate optical setups, possibly by using the Texas Instruments DLP technology.

Therefore, another experiment is designed to illustrate the sub-exposure image recovery with the FDMI technique. The experiment is conducted using the DMD on a TI Lightcrafter. The benefit of using DMD as opposed to LCD is that we have higher switch frequency and spatial resolution, no light loss at on-state, and zero light leak at off-state. With an LCD based system, we cannot have a zero light leak off-state or 100% light transmission during on-state. With DMD, we can implement high-speed pattern sequences with minimal transition. The recovered image, shown in figure 1.13e, is still suffering from some degree of artifacts, which is due to the fact that there is aliasing and that some high frequency components were cut off by the filters as shown in figure 1.13c. As seen in figure 1.13c, the Fourier domain is not used as efficient as possible. We can incorporate other sub-exposure gratings, at different frequencies and angles, and capture

more sub-exposure images. We can include an optical low-pass filter into the system to ensure band-limitedness and prevent aliasing.

CHAPTER 2.
2D FRAGMENTED IMAGE REASSEMBLY

2.1 Introduction

Fragmented Image reassembly is to reconstruct a 2D image back to its original state, after it is being torn or damaged for various reasons. Being able to accomplish such a task with computers will save us the human labor and time. Analyses show [1] that computers will be able to carry out such a task much faster than humans can do. Many different fields can benefit from solving such a problem. For example, forensic experts can save substantial amount of time and energy if the task of reassembling a piece of evidence that has been torn apart or shredded as an attempt to destroy the evidence can be handled automatically. Another application is for archeologists trying to reassemble pieces of artifacts they find through excavation, especially when the pieces are almost flat and smooth, and can be approximated as 2D image fragments. The reassembly process can be faster and more accurate if the task is done automatically by a program compared to when it is done manually.

Existing automatic image reassembly algorithms can be generally categorized into two main groups: color-based approaches and geometry-based approaches. In color-based approaches, the color information of the image fragments are used to guide the reassembly process. This approach can be sensitive to noisy pixel values and also may fail if there exist a few image fragments with similar texture close to the border areas. Geometry-based approaches, on the other hand, use the shapes of image fragments and their borders to find the proper matches. These methods can have big time complexities; and not taking advantage of the fragment's color information can be another drawback of this type of approaches.

There is another way to look at automatic image reassembly algorithms. Some algorithms operate on the pixel level to guide the reassembly process. These methods are also sensitive to noise. Others reassemble fragments by treating them as 2D geometric regions. Therefore, the time complexity will reduce for the latter approaches compared to the other category.

The key challenging issue in solving 2D image fragment reassembly problems is finding reliable matchings between different pieces. Due to small to no overlap between relevant image fragments in this type of reassembly problem, as opposed to well-studied image processing problems such as panorama creation or structure-from-motion where two adjacent frames usually demonstrate significant overlap, finding a reliable match between fragmented pieces is highly challenging. Therefore, it is essential to have effective pairwise matching algorithms, either using the fragments' geometry or color information to guide us through the inter-fragment alignment process. On the other hand, to avoid fully relying on the challenging and usually not so reliable pairwise alignments, it is critical to formulate our approach in the form of a pipeline and design an effective strategy to globally optimize a groupwise alignment which prunes and refines the pairwise alignments computed previously by maximizing their mutual consistency.

Our idea is to combine the positive aspects of different approaches mentioned above. We propose a three step composition pipeline as illustrated in figure 2.1. (1) Artifacts removal: each image fragment will be scanned and digitized before it is used as an input. The scanning process often creates undesirable artifacts, especially on and around the border area. A preprocessing step is introduced that ensures a proper execution of all next steps. (2) Pairwise matching detection: an effort to find potential matching between pairs of

image fragments. The pairwise matching step of the algorithm runs without global understanding of the layout of the other fragmented pieces. Therefore, some alignments suggested by pairwise matching step can be incorrect. (3) Global matching: a global matching is adopted to select correct alignments offered by the previous step, by optimizing the mutual consistency of alignments of multiple pieces together.

The Main contributions of this paper are as follows:

- We develop a necessary artifact removal preprocessing algorithm to eliminate noisy boundary tissues and smoothen the border of each image fragment to reduce noise (used in 1st step of the pipeline).
- Using correlation detection techniques based on deep neural networks, we identify potentially neighboring fragments to effectively reduce the search space for pairwise matching detection step (used in 2nd step of the pipeline).
- Effective partial matching between two image fragments is essential to the process. We develop a polygonization algorithm to model image fragment's boundary shape feature. Then, we formulate the pairwise matching as solving a novel variant of Longest Common Subsequence (LCS) problem. Compared with existing pairwise matching techniques, this algorithm improves the time complexity as well as robustness in handling images with possibly noisy values on borders. This algorithm effectively utilizes both geometric and color characteristics of image fragments.

2.2 Related Work

Finding a relative transformation between adjacent image fragments is the essence of most proposed automatic reassembly algorithms. Different authors utilize different measures to investigate a pair of image fragments for any possible matches. The two

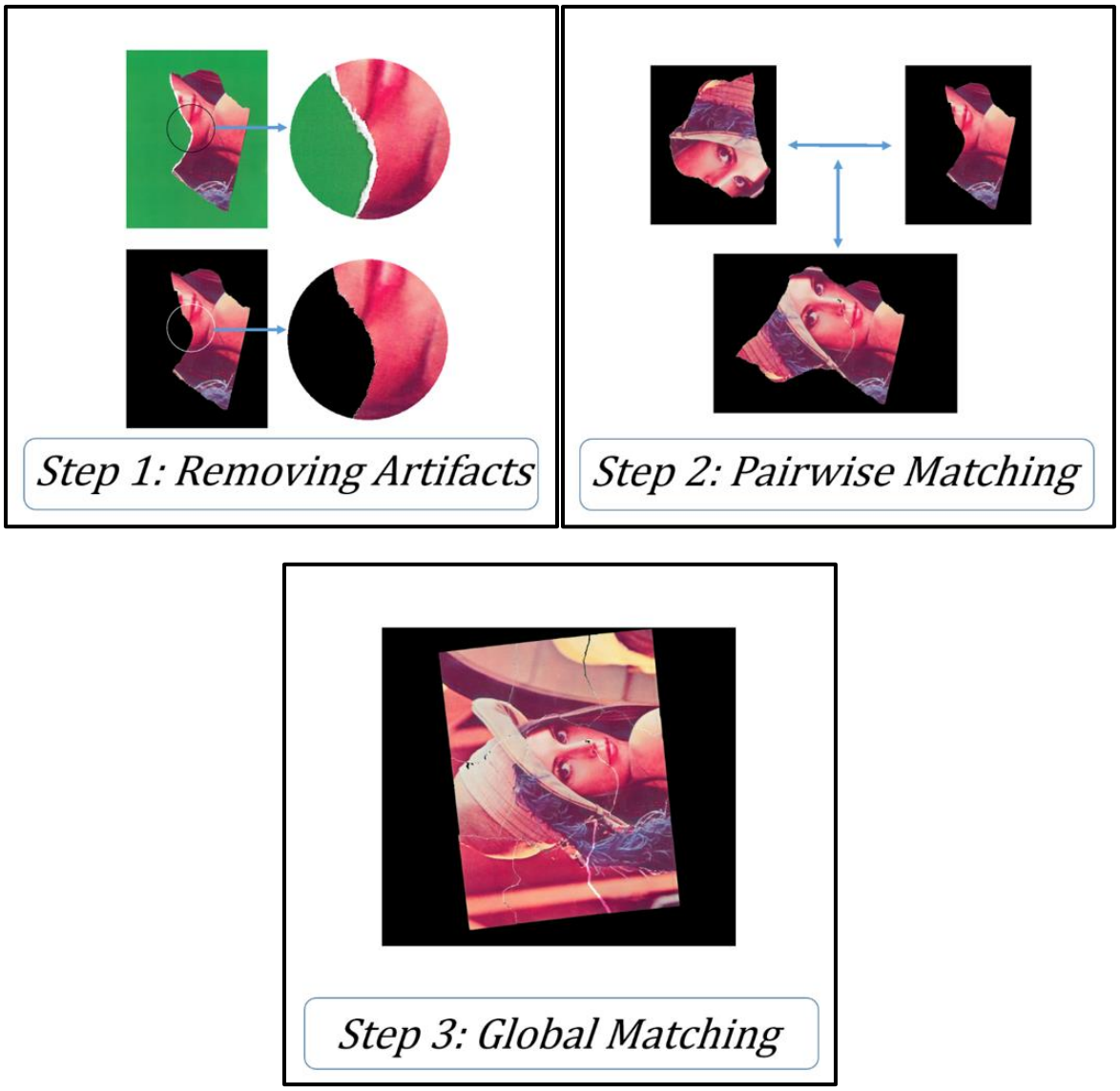


Figure 2.1: Three step pipeline
(a) Artifacts on and around the border area removed, (b) Pairwise matching, (c) Final result after groupwise matching

main approaches are based on color properties of the pixels on and around the border areas as well as geometry properties of the image fragments such as shape and curvature of the borders.

In color-based approaches, the color information of the image fragments are used to guide the reassembly process. Tsamoura and Pitas [1] form a list of pairs of image fragments that are likely to be adjacent. To build a fragment's descriptor, they use a neural

network based color quantization method and a novel approach to color histograms based on what Cinque et al. proposed in [10], called spatial-chromatic histograms, which not only considers color information but also considers spatial distribution of color. Authors in [1] present an algorithm based on LCS for identifying matching partial contour curves for each pair of fragments. Their approach to fragment contour matching is based exclusively on pixel color information on the contours. As a result, it loses some efficiency and is sensitive to noisy pixel values on the borders. In [11], the main focus is on pairwise matching and the authors also use a matching approach based on LCS. They try to find the largest common substring with the similar color and curvature information. In [12], the authors focus on the task of matching two image fragments only, using the information extracted from the outlines and from the color contents of the fragments. In [32], the texture of a band outside the border of pieces is predicted by inpainting and texture synthesis methods proposed in [33]. An FFT-based registration algorithm is then utilized to find the alignment of the fragment pieces. Color-based approaches can be sensitive to noisy pixel values and also may fail if there exist a few image fragments with similar texture close to the border areas.

Geometry-based methods, on the other hand, rely on analyzing the shape of image fragments and their borders to find the proper matches between adjacent pieces. These methods are especially effective in aligning fragments when their contour shapes are irregular. In geometry-based approaches, fragment borders are usually modeled as a 2D curve. Some use polygon approximation of the curves and some use local curvature information and shape features of the curves to match image fragments alongside their boundaries. Justino, Oliveira and Freitas in [4] solve the problem of reconstructing

shredded documents by first doing a polygonal approximation of the borders to reduce possible complexity of the boundaries. Then relevant features of each polygon, i.e. angle of each vertex with respect to its two neighbors and distances between each vertex and its neighbors, are extracted to lead the matching. In [5], a shape feature, referred to as the turning function, is calculated and used to investigate the matching of fragment pairs. Wolfson [14] finds the longest curve subsection being shared between polygonised fragment borders through geometric hashing. Its pairwise matching scheme lacks flexibility when it comes to allowing a deletion or mismatch of a pair of segments among a series of matching pairs of segments. [14] does not introduce a global matching step and the main focus of the proposed method is on pairwise matching. Zhang and Li in [6] combine both color-based and geometry-based methods and suggest approximating the border with a polygon. Instead of working on pixel level, the alignment is found alongside one of the polygon sides, therefore, reducing the effect of noise on pixel colors and taking the shapes of the image fragments into account also. In the pairwise matching step, an exhaustive search is used which is affecting the efficiency of the proposed method considering the large search space of the pairwise matching step. A global matching scheme is then proposed in [6] as solving a maximal compatible edge set from a given graph to reconstruct the original image from calculated local matches. The polygonization, in general, could be globally affected by the initial pose of the image fragments and the approximation error bound, and hence, is not completely rotation-invariant and will affect the subsequent matching and reassembly results. Local curvatures of pixels on the contour curve as the fragment's descriptor, on the other hand, will be rotation-invariant but is sensitive to geometric noise on the image boundary, which is often inevitable.

Based on an algorithm Smith and Waterman propose in [8], some algorithms [3, 11] try to find a common subsection between two neighboring border curves. Based on color values or curvature values at pixels they suggest a sequence of pixels as a possible common border subsection. In [3] the boundaries are represented by shape feature strings. A shape feature representing each pixel in the string is an average of the curvature values of the neighboring pixels. A method based on LCS detects similar substrings between two image fragments at progressively increasing scales of resolution. A similar approach is taken in [11] by Kong and Kimia. They try to find the largest common substring with the similar color and curvature information and their main focus is on pairwise matching. They use dynamic programing to create a coarse alignment on the reduced version of the borders. Then, they use dynamic programming again to get a fine-scale alignment. However, all the LCS-based methods mentioned above operate on pixel level and do not take advantage of polygonization technique to estimate a border curve as a polygon. The algorithms operating on pixel level are sensitive to noisy pixel values on the border. In addition, their time complexity can be big due to the large number of pixels on each border.

Although many 2D matching algorithms [36, 42, 45, 48, 49] and 3D matching [37, 47, 69, 70, 71, 72] algorithms have been proposed in related literature (some surveys can be found in [73, 74]), this topic remains challenging and there is, still, room for improvement.

2.3 Methodology

To reassemble 2D image fragments back into their original form, a wide range of approaches are being introduced in existing literature. In this work a novel method is devised in 3 steps.

2.3.1 Removing Artifacts

Each image fragment has to be digitized before it can be used as an input. A knowledgeable choice for background color during scanning process can enhance future steps to a great extent. The best possible choice for background color is a color that not only does not exist in the original image but also is furthest from all the existing colors. The color properties and color histograms of all image fragments are investigated to this end. This will facilitate segmentation and border extraction during next steps.

As a result of the scanning process, there will be scanning artifacts such as shadow around the borderlines of each image fragment. Also, before processing an image and after the foreground of that image is segmented, there will be some undesirable narrow white regions attached to the main body of the image, i.e. paper tissues. The paper tissue artifacts could have happened as the original image was being torn apart. These artifacts will affect both the geometry and color properties of image fragments on and around the border area and, unless removed, will result in noisy borders and, therefore, will cause the algorithm to fail eventually. To eliminate these artifacts, the idea is to start with a white-colored pixel on or close to the border and remove it from the foreground. Next, any white-colored pixels on the neighboring area of the first pixel are picked and removed from the foreground. And the same process is repeated until there are no white-colored pixels left on the neighboring area of the pixels that were picked last. Therefore, the tissue will be removed but the main body of the foreground will remain intact. After removing the tissue, some unconnected components might still remain outside of the main body of the image which will be eliminated using morphological operations [2]. Figure 2.2 depicts a zoomed in version of the result for better illustration.

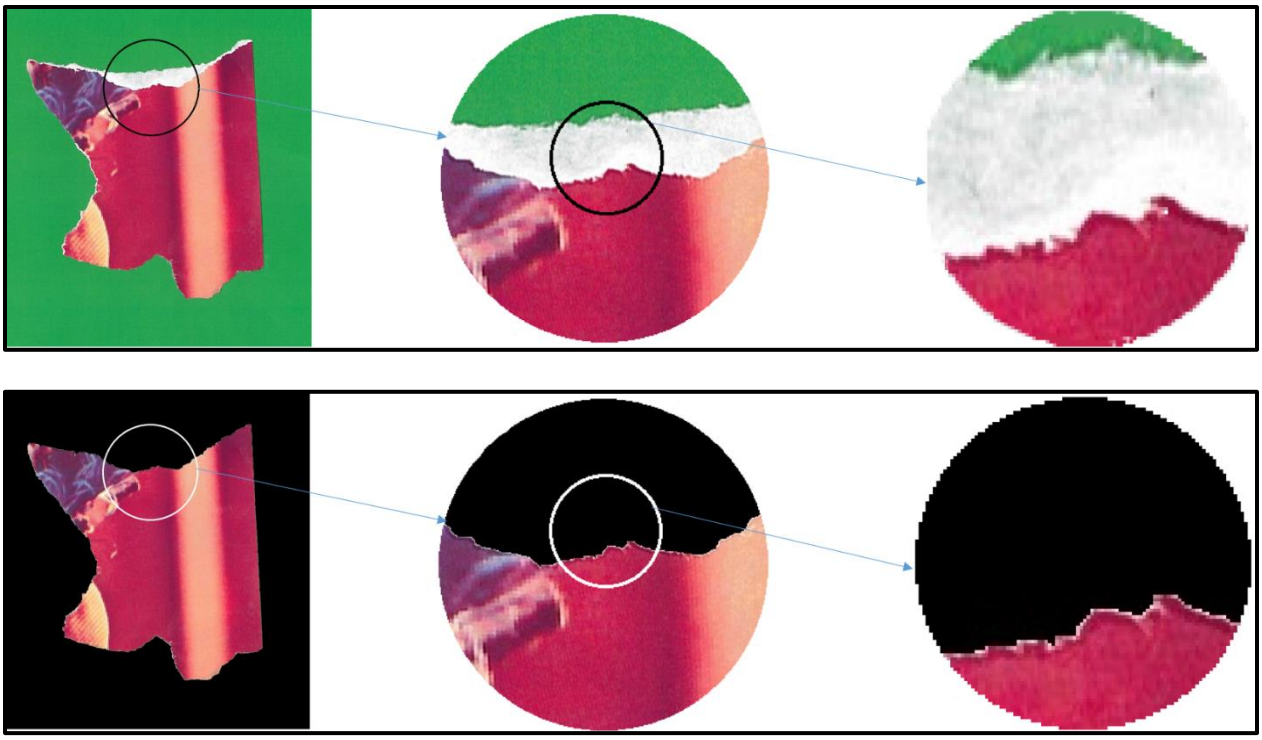


Figure 2.2: A demonstration for first step
(a) Artifacts on and around the border area, (b) Artifacts removed

The border extracted for each image fragment as a curve at this point carries many noise-like nuances. This curve, B, can be represented as an n-dimensional vector, where n is the number of pixels on B:

$$B = [p_1, p_2, \dots, p_n] \quad \text{where} \quad p_i \in \mathbb{R}^2 \quad \text{for} \quad \forall\, i \in \{1, \dots, n\} \tag{2.1}$$

Failing to smoothen the boundary curve can cause these noise-like nuances to be mistaken as feature points. Detecting the correct feature points is vital to our algorithm because correct polygonization depends on it. The smoothened version of B can be formulated as the solution to the following optimization problem, namely $X = [x_1, x_2, \dots, x_n]$ where $x_i \in \mathbb{R}^2$:

$$X = \operatorname{argmin}\left(\sum_{i=1}^{n-1} \| x_{i+1} - x_i \|_2\right), \quad \text{s.t.} \quad x_i \in \text{Sleeve}_B \quad \text{for} \quad \forall\, i \in \{1, \dots, n\}, \tag{2.2}$$

where $Sleeve_B$ is the small expanded area around the original border curve B. As a result, the noise on the curve will reduce and the border will become smooth. Figure 2.2 depicts a zoomed in version of the result for better illustration, once before paper tissue and noise on the border are removed and once after.

2.3.2 Pairwise Matching

The goal of pairwise matching step is to suggest border subsections on a pair of image fragments that is possibly common between the two of them. Consequently, there will be a transformation that best fits the image fragments alongside the identified common border subsection. This step of the algorithm runs for each pair of image fragments without global understanding of the layout of the other fragmented pieces. Therefore, this step suggests some correct and some incorrect alignments. The global matching (to be discussed later) will detect and eliminate incorrect alignments offered by pairwise matching step.

One major approach to represent the border of an image fragment is to describe it through geometric information like curvature values or color information at each pixel on the border [1, 3, 5, 11]. One main drawback for this way of describing an image fragment border is that it is sensitive to the noise on the border that can be caused as the original image was being damaged or torn into pieces and is unavoidable in real life scenarios. The other main drawback is that describing a border at pixel level captures only local information and will cost us overall geometrical information that is embedded in the shape of a piece as a whole. Therefore, a more efficient way of representing a border is to approximate it using a polygon. Therefore, instead of working on pixel level, those approaches using polygonization to approximate the border curve will look for alignment

alongside one of the polygon sides [4, 6, 14]. They need to employ an exhaustive search method to investigate the alignment of each side on one polygon against any other side on the other polygon in an effort to find the best alignment [6]. Zhang and Li [6] suggest to approximate the border with a polygon. Instead of working on pixel level, the alignment is found alongside one of the polygon sides. Therefore, not only the effect of noise on pixel colors is reduced but the shapes of the image fragments are taken into account also. On the other hand, each time only two sides, one from each polygon, will be investigated as the possible common border subsection that the two image fragments can be sharing. In other words, each side is investigated alone and separated from all the other sides of the polygon it belongs to. To rank these possible alignments against one another, a scoring scheme is required. An alternative version of Iterative Closes Point algorithm [7,13] is proposed to calculate the score for each alignment, which can be time consuming to evaluate.

But the novelty of our idea is based on the observation that if two image fragments are adjacent, their simplified polygons usually share more than only one side. Therefore, a more effective method to search for a common border subsection between two polygons approximating two fragment border curves is by searching for a sequence of matching sides, rather than considering one side at a time. We can formulate finding the best fitting common border subsection between two image fragment border curves as finding the longest subsequence of sides shared between the two polygons approximating their respective curves.

We propose to represent a border as a string for which each character describes one of the sides of the approximating polygon by bearing color and geometrical information of

that line segment. This will not only capture the local information, but will also help us take into account some properties of the overall geometrical shape of the polygon such as angles between two consecutive sides. Detecting the matching portion of two potentially neighboring borders, then, can be formulated as finding the Longest Common Subsequence (revised LCS) between the two strings describing the corresponding approximating polygons. Smith and Waterman [8] propose a solution to a similar problem based on a dynamic programming approach.

The goal for the pairwise matching step can be stated as calculating a 3×3 rigid transformation, $T_{i,j}$, which aligns i_{th} and j_{th} image fragments alongside their potential common border subsection. This goal is achieved through the following steps. The input to this part of the algorithm is a pair of image fragments after removal of the artifacts as explained before. Each boundary curve C_i is partitioned into a sequence of curve segments, each of which is flat and with uniform color and can be approximated by a line segment. Each curve segment in C_i will be denoted as $S_{i,k}$. The output of this step is the best possible sequence of pairings between $S_{i,k}$ and $S_{j,h}$. Finally, a single transformation that aligns the pairs is calculated. Our pairwise matching step is formulated in three steps.

a) Correlation detection.
b) The border extraction and polygonization using revised DP.
c) Detecting potential alignment based on revised LCS.

2.3.2.1 Correlation Detection

The key observation here is that most natural images display the same structure and color distribution in an area compared to other areas of the same image. Therefore, a pair of image fragments that are in fact neighbors are most likely to share same structure and

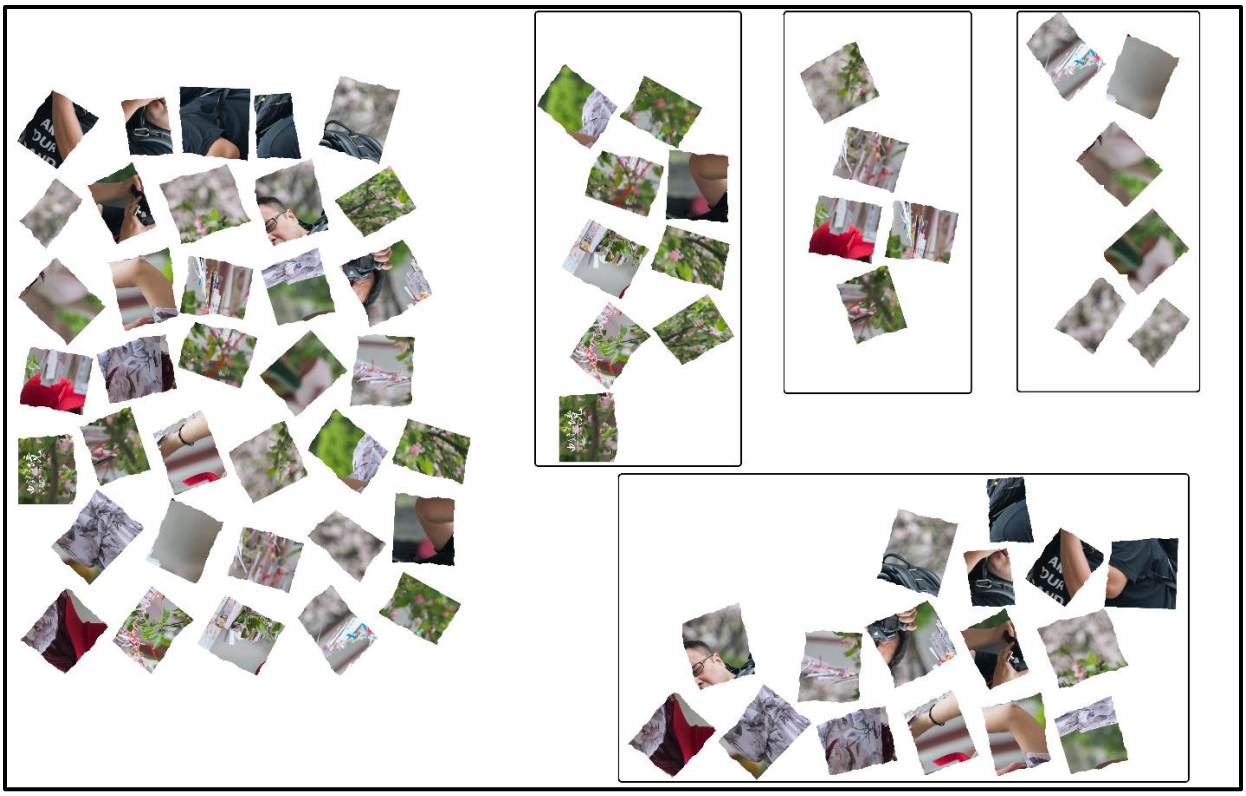


Figure 2.3: Results of correlation detection step on an input data set with 36 pieces. The fragments in each category are more likely to be neighbors.

color distribution as opposed to a pair that are not neighbors. Being able to effectively identify adjacency relationship of fragments, and therefore group together potentially adjacent pieces among all the fragments is highly desirable in reassembly, especially when the number of pieces is big and pairwise alignment is difficult. The reason is that this will reduce the search space for both pairwise and groupwise matching by limiting the number of pairwise matches that need to be computed and optimized in groupwise step. In other words, the need to consider each image fragment with all other image fragments exhaustively in the pairwise matching step is limited only to the ones that are most likely in the neighboring area.

We introduce a fragment adjacency estimation algorithm, which is based on [34]. This method is based on the convolutional neural network (CNN) to suggest adjacent pieces

and filter out irrelevant fragments. A typical CNN consists of many layers of convolutional operations. Each convolutional layer is associated with a collection of image filters. Given the i_{th} layer, feature maps corresponding to that layer are extracted through one or more steps of convolution of previous layer's feature maps with i_{th} layer's image filters.

For each fragment, a style matrix, S, is calculated which encodes the style and structure of that image fragment. Assuming we have M feature maps at a specific layer of our CNN, S is an $M \times M$ matrix built upon the feature maps of this layer:

$$S_{i,j} = \frac{F_i \,.\, F_j}{|F_i| \,.\, |F_j|}, \quad \text{s.t. } F_i, F_j \text{ are flatten vectors of } i_{th}, j_{th} \text{ feature maps} \tag{2.3}$$

We then flatten the style matrix for each image fragment into a vector and use their pairwise cosine similarities to construct a neighboring matrix of size $N \times N$ assuming we have N image fragments in our problem. Image fragments are then categorized into groups of similar style based on the calculated neighboring matrix and hierarchical k-medoids clustering method.

Figure 2.3 illustrates the result of this step of the algorithm on an input data set with 36 fragmented pieces. The input pieces are divided into 4 categories of similar structure and color distribution.

2.3.2.2 Border Extraction and Polygonization Using Revised DP

For each selected pair of image fragments, the first task is to extract the border loop on each image fragment as a curve and approximate it as a polygon. Polygonising each contour curve means it is partitioned into a sequence of curve segments, each of which flat enough to be approximated by a line segment.

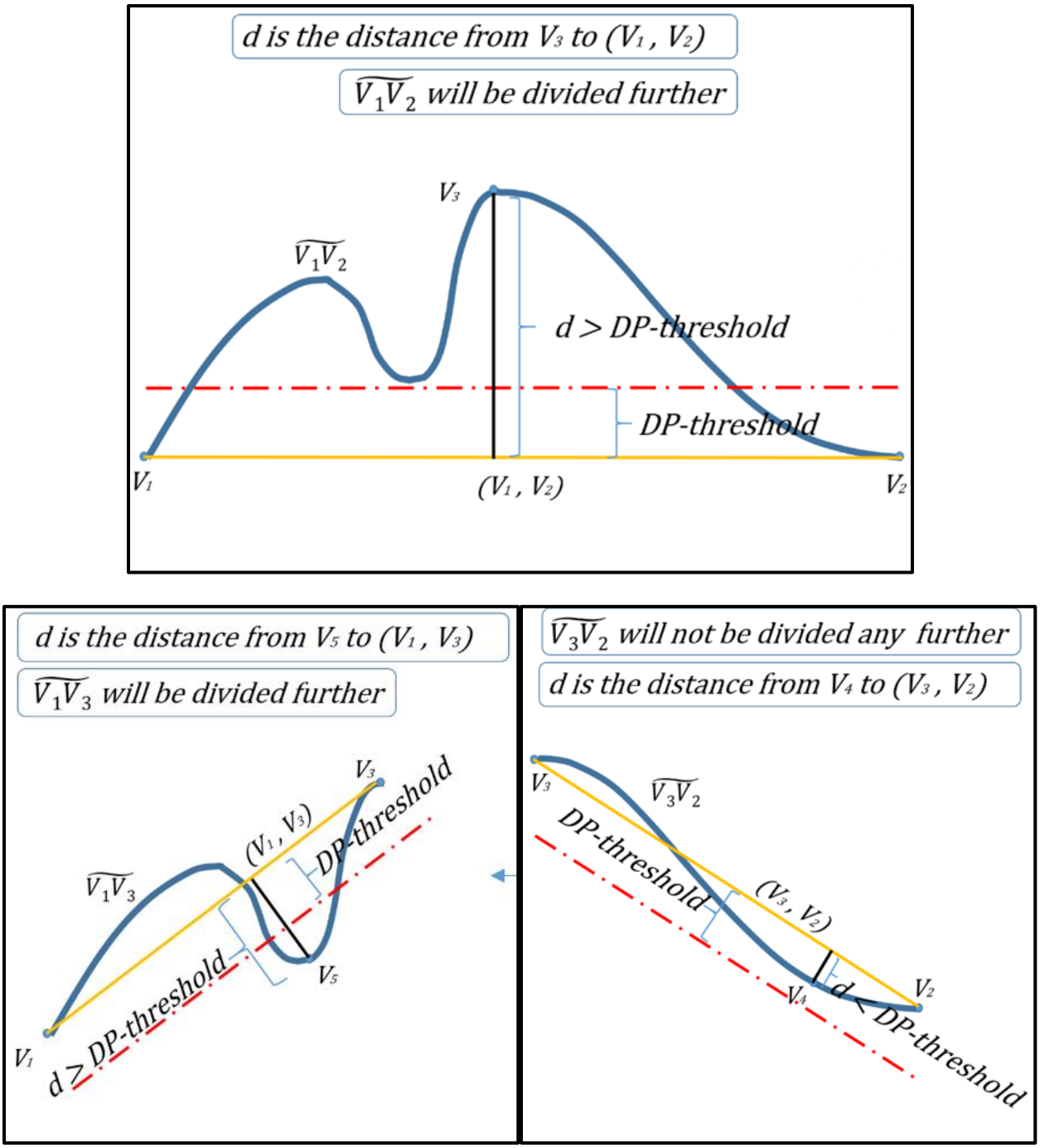


Figure 2.4: An illustration for DP algorithm
(a) First curve segment to start the partitioning process with, (b) This sub-segment will be partitioned to $\widehat{V1V5}$ and $\widehat{V5V3}$ in the next step, (c) This sub-segment is already almost flat and there is no need to partition it any further.

A classic approach to decomposing an open curve into smaller and flatter segments is through Douglas-Peucker (DP) algorithm [9]. For a curve segment denoted as $S = \widetilde{V_1V_2}$, where V_1 and V_2 are the starting and ending points of the segment, DP algorithm first uses a line segment connecting V_1 and V_2 to approximate the curve, denoted as (V_1, V_2) as shown in figure 2.4a. The distance from each point V_i on the curve to line segment (V_1, V_2) is calculated. Assuming there exists a point on the curve, V_3, with the largest distance to (V_1, V_2) while its distance is longer than a predefined threshold, the segment $\widetilde{V_1V_2}$ is decomposed into two smaller segments, $\widetilde{V_1V_3}$ and $\widetilde{V_3V_2}$. The DP algorithm then runs recursively on $\widetilde{V_1V_3}$ and $\widetilde{V_3V_2}$ until no curve segment can be further decomposed. See figure 2.4b and figure 2.4c. Now, Let us assume C_i and C_j are the boundary curve loops extracted for the pair of image fragments at hand. To segment a closed loop, like C_i or C_i, DP cannot be applied directly since DP applies to open curves which have starting and ending points. To start the process, a closed loop like C_i needs to be initially segmented into two or more open curves.

Noticeably, the polygonization of a looped curve could be globally affected by the initial pose of the image fragment, the approximation error bound and the noise on the border and hence, could affect the subsequent matching and reassembly results. Therefore, selecting the correct initial vertices is essential for a successful polygonization. To select the best possible candidates as initial vertices, we pick the points on the border with curvature values larger than a threshold as corner feature points. The DP algorithm, described above, can then run on the initial segments respectively. As a final result, the boundary loop C_i will be partitioned into a sequence of small and flat curve segments $S_{i,k}$, the k_{th} of such curve segments on C_i, for example.

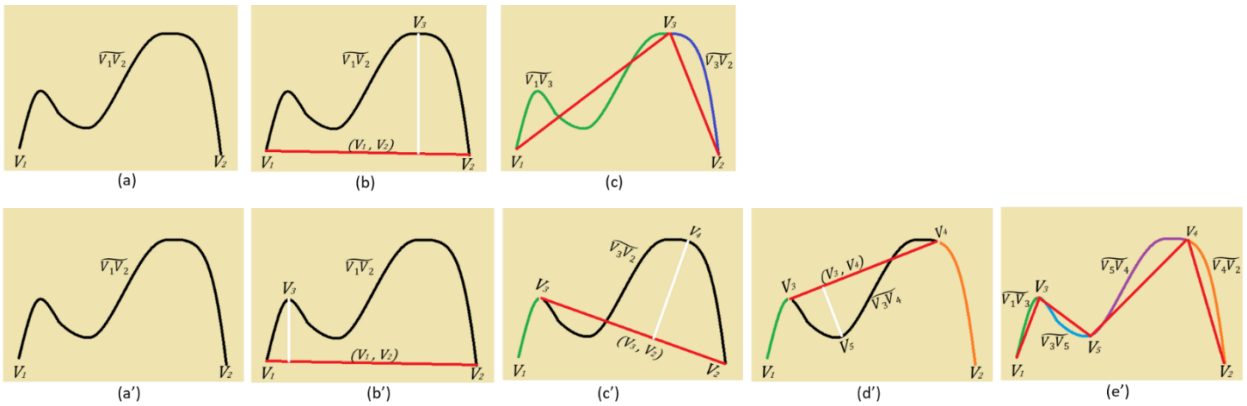


Figure 2.5: The importance of incorporating the curvature values into the segmentation process. (a, a') The original curve to be segmented, (b, c) The decision on where to break the curve is made, solely, based on the Euclidean distance each pixels makes with the line segment describing the curve, (b', c', d', e') The Euclidean distance a pixel makes with the approximating line segment is scaled based on the curvature value at that specific pixel. As a result, pixels with high curvature values are more likely to be picked as vertices during segmentation process. For example, the first vertex to be picked in (b') is V_3 due to its high curvature value, while there are points on the curve that have bigger Euclidean distance to the line segment (V_1, V_2).

To reduce the effect of the approximation error bound and the noise on the border in the polygonization process, we propose to scale the distance of each point on a curve $\widetilde{V_1V_j}$ to the line segment (V_i, V_j) by a factor reflecting the curvature value at that specific point on the curve. Figure 2.5 is devised for illustration purposes to emphasize the importance of including curvature value at each point on a curve into that curve's segmentation process through a synthetic example. Picking pixels with higher curvature values as vertices of the polygon, during segmentation process, is closer to the way a human would do such a task compared to the previous methods proposed.

Figure 2.6 shows how the polygonization of border curves are improved as a result of artifacts removal (first step in the pipeline) and applying revised DP as compared to applying just DP to the, rather noisy around the border, original border. This improvement will be very helpful during the next step in pairwise matching.

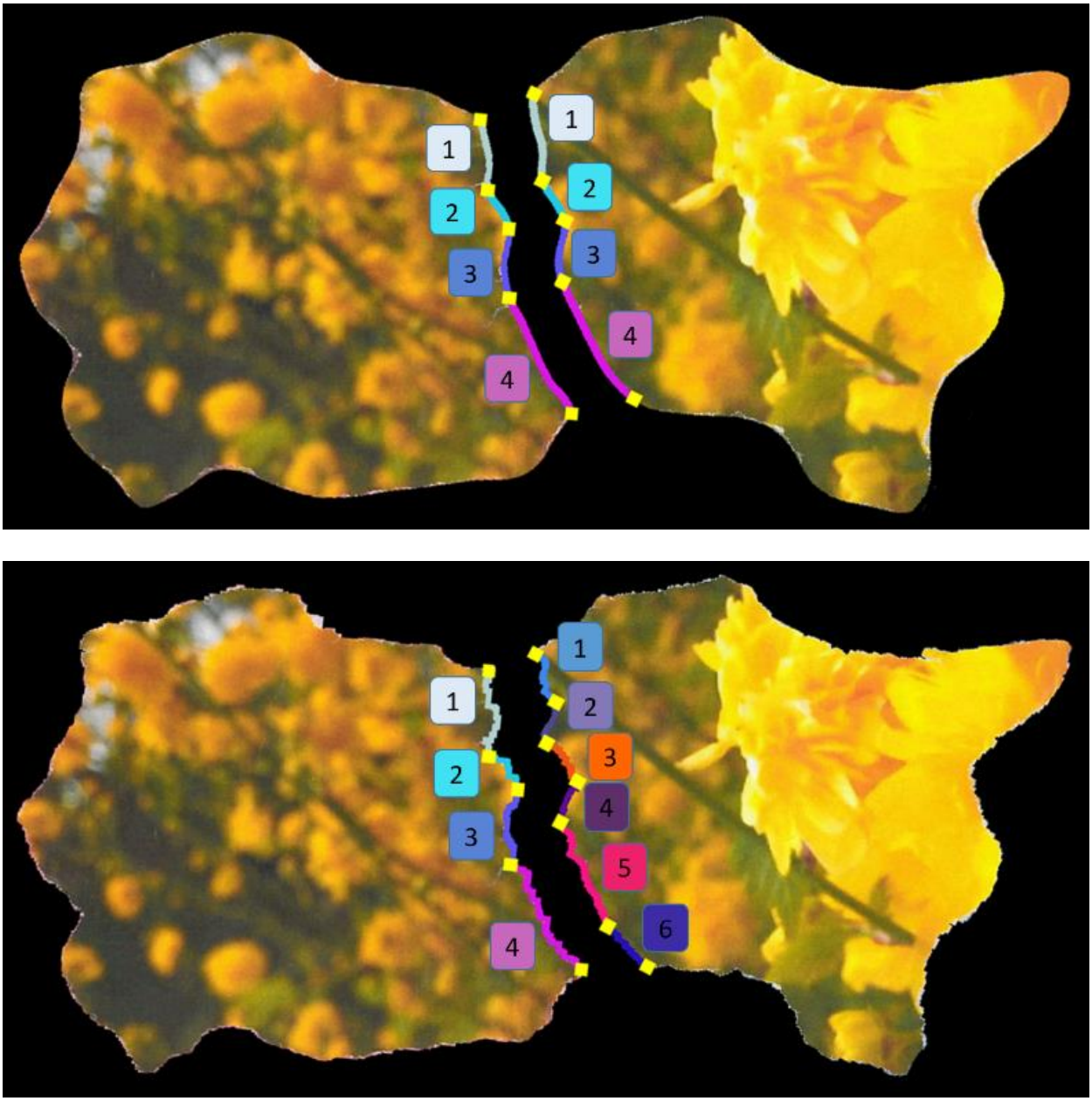


Figure 2.6: Comparing the result of (a) revised DP, (b) DP.

For two adjacent image fragments, there is a common subsection between their border loops. This common subsection is usually composed of more than one curve segment. The goal is to find the longest sequence of pairings between segments on the first border loop and the second border loop. To solve this problem, we revise the Smith-Waterman

algorithm [8], which uses dynamic programming to find the longest common subsequence (LCS) between two given strings. The relationship between Smith-Waterman algorithm and our solving of the problem will be discussed in more details in upcoming sections.

2.3.2.3 Detecting Potential Alignment Based on Revised LCS

After partitioning a boundary curve C_i into smaller curve segments, each segment $S_{i,k}$ on the boundary can be represented using a descriptor, $d(S_{i,k})$. This descriptor is formed base on the color and geometric properties of each curve segment and will be in a 4D (1D length plus 3D color) space. Later we will talk about this descriptor in more details. Considering each descriptor equivalent to a character, a boundary curve composed of a sequence of segments, can be represented as a string. The length of the string representing a boundary will be equal to the number of segments on that boundary. Finding the longest sequence of curve segments shared between two borders will be equivalent to the problem of finding the longest subsequence shared between two given strings.

Let's consider two long strings $C_i = d_{i,1}\, d_{i,2}\, d_{i,3} \ldots d_{i,n}$ and $C_j = d_{j,1}\, d_{j,2}\, d_{j,3} \ldots d_{j,m}$ where each string represents a border and $d_{i,k}$ for $1 \leq k \leq n$ and $d_{j,l}$ for $1 \leq l \leq m$ are two characters equivalent to descriptors representing k_{th} and l_{th} curve segments on C_i and C_j respectively. Based on Smith-Waterman algorithm we employ a dynamic programming approach to find a pair of substrings, one from each of two long strings, such that there is no other pair of subsequences with greater similarity. Through identifying the most similar subsequences, this approach introduces a matching between their characters. To make matching between characters flexible, jumping over a character in either one of the

subsequences and simply leaving it unmatched with any of the characters from the other substring is allowed.

This act is called a deletion and is justified when, doing so, there will be a prospect of being able to make more promising matches along the path as we go forward and create new matches. Approaching the problem from this point of view, the two identified subsequences do not have to be the longest identical substring that are being shared by A and B, but they only have to be the most similar ones among all possible choices.

For a matching proposed between any two subsequences, a similarity measure is calculated by accumulating points as we go along and create matches. The pair of substrings with the highest similarity measure is considered to be the desired pair with the desired matches between their characters. In a proposed matching, if a character is left unmatched, this causes some dissimilarity between the two substrings at hand. Therefore, a penalty is considered for such a case by adding a negative value to the accumulated similarity measure calculated up to that point. We represent this negative penalty by W. If two characters are matched, on the other hand, we represent the contribution of this match to the accumulated similarity measure by $\mathrm{Sim}(d_{i,k}, d_{j,l})$. If the match is a good one, $\mathrm{Sim}(d_{i,k}, d_{j,l})$ will be a positive value. On the other hand, if it was a bad match, $\mathrm{Sim}(d_{i,k}, d_{j,l})$ will be negative and accumulated similarity measure will be penalized for the dissimilarity being introduced by this bad match. Later, we will elaborate more on what we mean by a good or a bad match and how to properly calculate $\mathrm{Sim}(d_{i,k}, d_{j,l})$. Therefore, as we go along we have the option to (1) create a match between to characters, (2) create a mismatch between two characters, or (3) simply decide to jump over a character and leave it unmatched with any of the characters on the other

subsequence. Case 1 will reward accumulated similarity measure by increasing it, while case 2 and 3 will cause a penalty towards the accumulated similarity measure by decreasing it. The reason to take such penalties is the hope we have to be able to make more desirable matches ahead and therefore, make up not only for the negative penalties taken but even more.

To calculate the pair of subsequences with highest similarity measure, a dynamic programming approach is utilized. First, an $n \times m$ matrix called H is set up, where n is the length of C_i and m is the length of C_j. First row and column of H will be initialized to all zeros. Therefore,

$$H_{k,0} = H_{0,l} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m. \tag{2.4}$$

Preliminary values of H have the interpretation that $H_{k,l}$ is the maximum similarity of two subsequences ending in $d_{i,k}$ and $d_{j,l}$ respectively. In other words, $H_{k,l}$ is the similarity measure for the pair of subsequences ending in $d_{i,k}$ and $d_{j,l}$. These values are obtained from the relationship

$$H_{k,l} = \max\{H_{k-1,l-1} + \mathrm{Sim}(d_{i,k}, d_{j,l}), H_{k-1,l} - W, H_{k,l-1} - W, 0\} \tag{2.5}$$

$$\text{for all } 1 \leq k \leq n \text{ and } 1 \leq l \leq m.$$

The formula for $H_{k,l}$ is reached at by considering the possibilities for ending the substrings at any $d_{i,k}$ and $d_{j,l}$.

a) If $d_{i,k}$ and $d_{j,l}$ are matched, the similarity measure is $H_{k-1,l-1} + \mathrm{Sim}(d_{i,k}, d_{j,l})$.

b) If $d_{i,k}$ from C_i is not matched with any of the characters in C_j, the similarity measure is $H_{k-1,l} - W$.

c) If $d_{j,l}$ from C_j is not matched with any of the characters in C_i, the similarity measure is $H_{k,l-1} - W$.

d) Finally, a zero is included to prevent calculated negative similarity measures, indicating no similarity up to $d_{i,k}$ and $d_{j,l}$.

The pair of substrings with maximum similarity is found by first locating the maximum element of H. The other matrix elements leading to this maximum value are then sequentially determined with a trace-back procedure ending with an element of H equal to zero. This procedure identifies the subsequences as well as produces the corresponding matches between their composing characters. The pair of substrings with the next best similarity is found by applying the trace-back procedure to the second largest element of H not associated with the first trace-back. Assuming we have already calculated $\mathrm{Sim}(d_{i,k}, d_{j,l})$ for all $1 \leq k \leq n \text{ and } 1 \leq l \leq m$, review of the algorithm for this part is given in figure 2.7.

As mentioned above, each string is another way of representing a border if we think of characters in that string as equivalents to curve segments in that border loop. The ultimate goal of this step is to calculate a transformation that aligns the two borders alongside the identified pair of curve segment sequences. Among proposed matches for an identified pair of curve segment sequences, those specific curve segments that are left unmatched due to a deletion operation will be excluded from the final transformation calculation. For each match, or possibly mismatch, between two curve segments there exists a transformation that aligns the two borders alongside those specific curve segments and those specific curve segments only. But we are looking for just one transformation that after transforming the borders, aligns the most possible number of matched curve

// n is the number of curve segments on the border of image fragment C_i

// m is the number of curve segments on the border of image fragment C_j

considering a fixed i and a fixed j

for all $0 \leq k \leq n$

    $H_{k,0} = 0$

for all $0 \leq l \leq m$

    $H_{0,l} = 0$

for all $1 \leq k \leq n$

    for all $1 \leq l \leq m$

        $H_{k,l} = max\{ H_{k-1,l-1} + Sim(d_{i,k}, d_{j,l}), H_{k-1,l} - W, H_{k,l-1} - W, 0 \}$

produce the corresponding matches by starting from the maximum element of H and tracing back to an element of H equal to zero

Figure 2.7: Review of the algorithm for finding matching curve segments using LCS

segments next to one another. For two adjacent image fragments, the transformations corresponding to the good matches have to be consistent and will end up being the same correct transformation we are looking for. On the other hand, a transformation corresponding to a mismatch will not be consistent with either that one desired transformation from good matches or other transformations from other mismatches. In order to exclude the incorrect transformations introduced by mismatches from the final transformation calculation process, we list all the transformations introduced by all matches and mismatches and simply pick as our desired transformation the one that repeats the most, i.e. the mode of the distribution.

While looking for the most similar pair of curve segment sequences and the possible matches between them, the fact that some curve segments can be left unmatched due to a deletion operation, fits the specific layouts of our problem. The reason is that partitioning a border into smaller curve segments is a difficult problem and is an algorithm that has

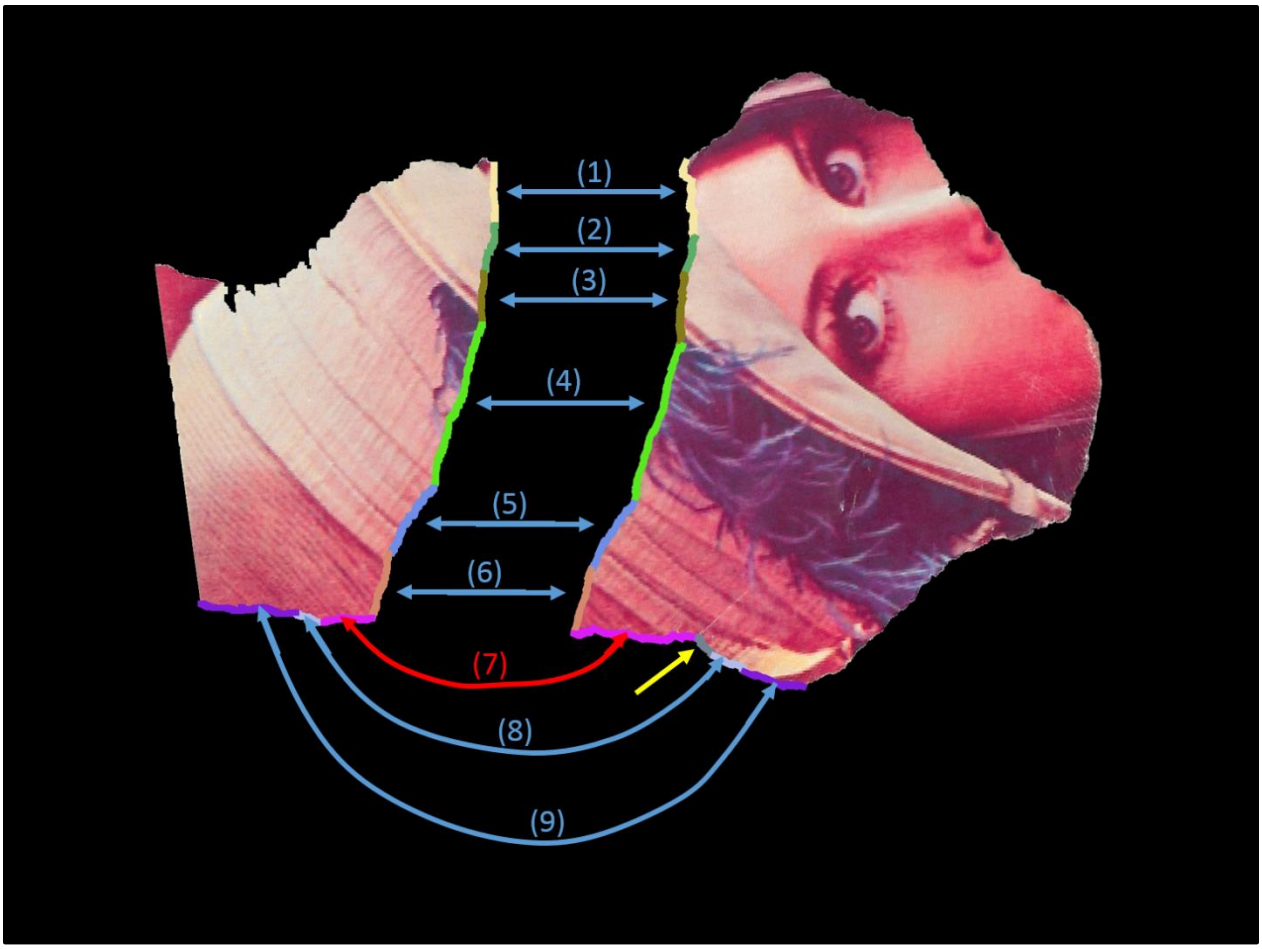


Figure 2.8: The result of revised LCS for two image fragments.
The matched segments are color coded. (All matched segments marked with blue arrows are good matches. The one short curve segment marked with a yellow arrow is showing a case of deletion due to wrong partitioning of the border. Match (7), marked with a red arrow, shows a case of a bad match. The two curve segments have different lengths. These two choices, i.e. deletion and mismatch, are being made for the benefit of matches (8) and (9) that are coming after them.) When calculating the final transformation between the two image fragments, the transformations introduced by matches (1)-(6) will all be consistent and the transformations introduced by (7),(8) and (9) will be considered as outliers and will not have any effect on the calculation.

room for improvements. For some instances, a curve segment on the first border that is also being shared with the second border is partitioned one step further on the second border and forms two curve sub-segments. In such a case, deletion provides us with the opportunity to jump over one of the two sub-segments without having to match it with any other curve segment on the first border and to keep trying to match curve segments that are further ahead. Figure 2.8 shows the most similar pair of curve segment sequences, one from each of two long strings, recovered using our version of LCS solution.

For each curve segment, $S_{i,k}$, on a border like C_i, we define a descriptor, $d(S_{i,k})$, based on the color and geometric properties of that curve segment. Assuming $S_{i,k}$ and $S_{j,l}$, for specific values of k and l, are the same curve segment shared by borders C_i and C_j respectively, the properties to form the descriptor should be chosen in such a way that $S_{i,k}$ and $S_{j,l}$ end up having similar descriptors. On the other hand, the descriptors should help us differentiate two curve segments that are not the same. For this reason, we decide on a descriptor in the 4D space. The descriptor $d(S_{i,k})$ for curve segment $S_{i,k}$ will be a 4-tuple value consisting of the length of $S_{i,k}$ and 3D color information representing $S_{i,k}$. To have only one 3D color value represent a whole curve segment, we have to average red, green and blue color channels for all the pixels on that curve segment.

$$d(S_{i,k}) =< \text{length}(S_{i,k}), \text{average}_{\text{red}(S_{i,k})}, \text{average}_{\text{green}(S_{i,k})}, \text{average}_{\text{blue}(S_{i,k})} > \qquad (2.6)$$

As mentioned begore, for a pair of curve segments $S_{i,k}$ and $S_{j,l}$ on borders C_i and C_j respectively, a $\text{Sim}(S_{i,k}, S_{j,l})$ will be calculated. The values calculated for $\text{Sim}(S_{i,k}, S_{j,l})$, for all possible values of k and l, affect the result of the LCS algorithm profoundly. We know that if $S_{i,k}$ and $S_{j,l}$ are similar, then their descriptors, $d(S_{i,k})$ and $d(S_{j,l})$, will be two equal values in the 4D space we have defined for descriptors. But if the two curve segments

are not similar their descriptors will fall far from one another in the 4D space. Based on this observation, we define $\mathrm{Sim}(S_{i,k}, S_{j,l})$ for two curve segments $S_{i,k}$ and $S_{j,l}$ to be 1 if their descriptors are equal. In case the descriptors are not equal, on the other hand, it means the two curve segments are forming a mismatch rather than a match. In such a case, $\mathrm{Sim}(S_{i,k}, S_{j,l})$ should end up having a negative value. This value is calculated dynamically based on the length of the curves that are being mismatched. $\mathrm{Sim}(S_{i,k}, S_{j,l})$ is not calculated dynamically based on the length of the curves because a match is as much valuable to us when it happens between two short curve segments as it is when it happens between two long curve segments, which is not the case with mismatches. In case of deletion, the penalty also known as W, is also calculated dynamically based on the length of the curve.

A shared border subsection between two adjacent image fragments is usually composed of more than one curve segment. The angle each curve segment on the shared border area of the first image fragment makes with its next and previous curve segments on that shared border area must be the same as the angles its counterpart makes on the shared border area of the second image fragment. The shape of the shared border area will not be the same on both image fragments geometrically if this condition does not satisfy. That is why, before $\mathrm{Sim}(S_{i,k}, S_{j,l})$ is set to 1 or negative value mentioned above, this condition is checked as well. Each one of the curve segments shared between the two borders are flat enough to be approximated by a line. These lines can be utilized to define angle between one curve segment and its next segment on the same border unambiguously. Figure 2.9 shows an illustration. A review of the algorithm to calculate $\mathrm{Sim}(S_{i,k}, S_{j,l})$ for two curve segments $S_{i,k}$ and $S_{j,l}$ is given in figure 2.10.

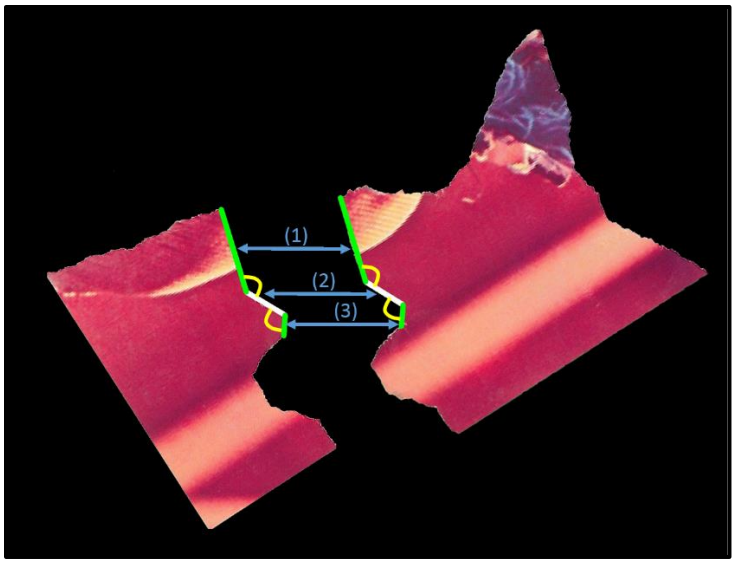


Figure 2.9: An illustration for the angle condition
The curve segment marked by number (2) is the match under investigation.

// n is the number of curve segments on the border of image fragment C_i

// m is the number of curve segments on the border of image fragment C_j

1 considering a fixed i and a fixed j

2 for all $1 \leq k \leq n$

3 calculate $d(S_{i,k})$ as described above

4 for all $1 \leq l \leq m$

5 calculate $d(S_{j,l})$ as described above

6 for all $1 \leq k \leq n$

7 for all $1 \leq l \leq m$

8 if $([d(S_{i,k}) = d(S_{j,l})]$ && $[angle(S_{i,k}, S_{i,k+1}) = angle(S_{j,l}, S_{j,l+1})]$ && $[angle(S_{i,k-1}, S_{i,k}) = angle(S_{j,l-1}, S_{j,l})])$

9 $Sim(S_{i,k}, S_{j,l}) = 1$

10 else

11 $Sim(S_{i,k}, S_{j,l})$ = negative value based on the lengths of $S_{i,k}$ and $S_{j,l}$

Figure 2.10: Review of the algorithm to calculate similarity value for two curve segments

The more accurately we describe two curve segments through their descriptors, the more accurately we will be able to tell whether they are similar or not. As a result, we can decide with more confidence whether two curve segments should form a match or a mismatch. It is for this reason that using better criteria in creating descriptors can lead to better final outputs.

During the first step of pairwise matching algorithm, the border of an image fragment is partitioned into curve segments based on DP algorithm. Increasing this DP-threshold value will result in a coarser partitioning of the border and decreasing it will result in a finer representation of the border. It is desirable to adaptively choose the DP-threshold according to the shape and resolution of the image fragments. A too small DP-threshold will cause the partitioning algorithm to become more sensitive to noisy border areas. As a result, smallest nuances on the borders will lead to creating new curve segments. On the other hand, a too large DP-threshold will cause the partitioning algorithm to lose some details on the border. Therefore, the curve segments will become very long, with deep curves which cannot be approximated with a line properly. Figure 2.11 depicts an image fragment and its extracted border. It emphasizes the importance of choosing the proper DP-threshold based on the resolution of input data set by showing the results of three different partitioning processes with different DP-thresholds.

The polygonization of the border is also affected by the choice of the starting point $V_1$and ending point V_2. However, when the DP-threshold is small and the curve subdivision is fine, the approximation result is still mainly decided by the intrinsic geometry of the border and is not sensitive against the selection of V_1 and V_2.

When calculating $\mathrm{Sim}(S_{i,k}, S_{j,l})$ for $S_{i,k}$ and $S_{j,l}$, as mentioned before, the curve segments are considered similar if their descriptors are equal. Otherwise, they will be considered a mismatch. In order to be able to tolerate a small level of noise, it is better to consider $d(S_{i,k})$ and $d(S_{j,l})$ equal as long as the Euclidean distance between them is smaller than

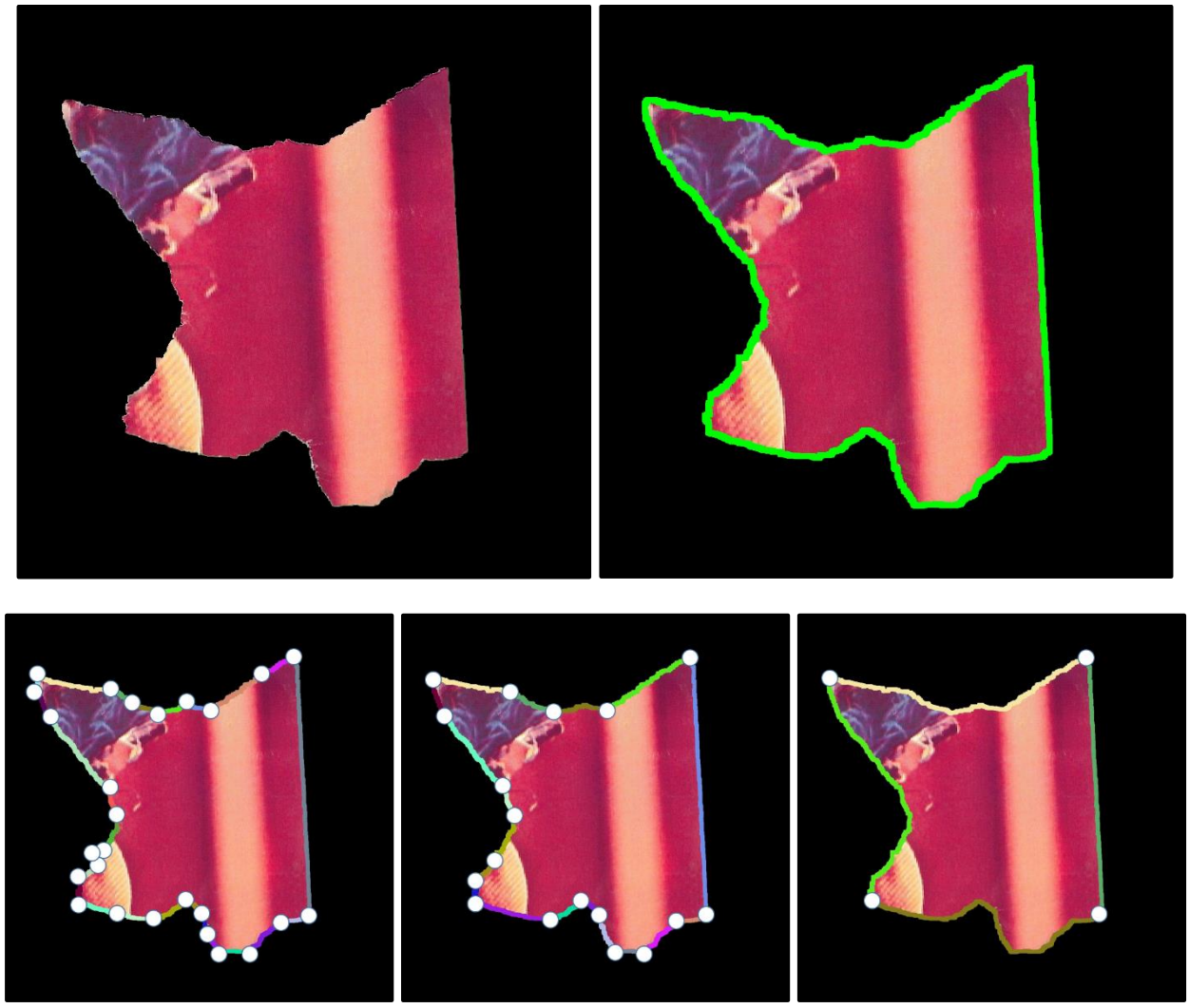


Figure 2.11: Boundary extraction and polygonization
(a) The original image fragment, (b) The extracted boundary contour, (c) Polygonization with DP-threshold equal to 5. The partitioning is too fine in this case. (d) Polygonization with DP-threshold equal to 20 and (e) Polygonization with DP-threshold equal to 150. The partitioning is too coarse in this case. (Random colors are being used to color curve segments for illustration purposes only)

a predefined threshold in the 4D space, i.e. they will be considered equal if they are close enough. It is also the case when comparing angles. The difference between two angles needs to be only smaller than an angle threshold for the two angles to be considered equal.

2.3.3 Global Matching

Pairwise matching step of the algorithm runs for all possible pairs of image fragments without global understanding of the layout of the other fragmented pieces. Therefore, it suggests one or more possible transformations between each pair of image fragments, among which only some can be correct and some others are incorrect. For the pairs of image fragments that are not adjacent, none of the proposed transformations between them are correct. On the other hand, for an adjacent pair of image fragments, one and only one of the proposed transformations between this pair is correct. We use a global matching step to detect and eliminate incorrect alignments offered in the previous pairwise matching step.

Global matching step, in fact, is used to complement the work of pairwise matching. This means the more sophisticated pairwise matching step we have, the less work we need to perform in the global matching step. If we develop a simple pairwise matching step on the other hand, we are simply pushing more work down the pipeline to be accomplished during the global matching step. For example, in [6], their relatively simple pairwise matching strategy returns many possible transformations between a pair of image fragments, and there is a need for a powerful global matching scheme to differentiate correct transformations from incorrect ones. Here, with a more effective pairwise matching, the obtained pairwise alignment is more reliable; and therefore, the need for a

list of possible transformations between a pair of image fragments is less urgent. Consequently, a simpler global matching algorithm usually suffices to reassemble the fragments using our pairwise matching results.

Zhang and Li in [6] formulate the global matching scheme as solving a maximal compatible edge set from a given graph. They propose a greedy algorithm to iteratively insert an alignment that is consistent with all existing selected alignments and has the highest accumulated matching scores with all the other pieces. In our framework, we also adopt this strategy. The reason is that, doing so, instead of having a list of length one for the potential transformations between a pair of fragments, we can handle a list with the top 2 possibilities as well. To increase the length of the list from one to two we can simply extract the next best pair of curve subsequences from the H matrix we developed while solving LCD problem and find the transformation that corresponds to that. Using global matching proposed in [6] gives us more assurance about the robustness of our method. At the same time, it does not increase the time complexity of our algorithm because we are still dealing with a very short list, i.e. length 2, compared to the list created by the pairwise matching step of [6]. It is worth mentioning the fact that when the length of the list of possible transformations is 1 for all pairs of image fragments, the global matching proposed in [6] will simply reduce to a best-first search strategy. It will lead to the correct final result only if there is high trust for that one transformation in the list to be the correct one.

2.4 Experimental Results

We perform multiple experiments to evaluate our assembly algorithm. In our experiments, we print out randomly selected digital images on papers. Most images, after being printed,

are about $20\text{cm} \times 25\text{cm}$. Then, we randomly tear an image into multiple pieces and scan each image fragment. The size of the fragments is ranging from about $4\text{cm} \times 4\text{cm}$ to about $7\text{cm} \times 7\text{cm}$. Our reassembly algorithm is then used to recompose the scanned digital image fragments.

The results in figure 2.12 demonstrate the effectiveness and robustness of our approach for real hand-torn images with increasing number of pieces. Most existing reassembly methods assume the torn image fragments are free from tearing artifacts like paper tissues and scanning artifact like shadows around the boundary of an image fragment. In some other cases this type of noise is handled manually through user interaction in a preprocessing step. In the case of our method, the results presented in figure 2.12 demonstrate the capability of our approach to handle tearing and scanning artifacts automatically.

We implemented the approach proposed in [6] for comparison purposes. See figure 2.13 for an illustration. The final results do not show any one algorithm's superiority over the other one. But the running time for our method was smaller than the running time for the algorithm in [6]. It took our method about 5 minutes to create the final output while it took the other approach around 20 minutes to create the same output. The reason for the difference in time complexity is explained rigorously later.

We have selected [1] method and implemented their proposed algorithm so that we can compare its performance with our method. [1] will fail on cases where the tearing or scanning artifacts on image fragments are not handled properly. Therefore in order to create a level ground, we created input datasets free of artifacts through simulation. The comparison is performed on cases with 9 pieces, 18 pieces and 36 pieces.

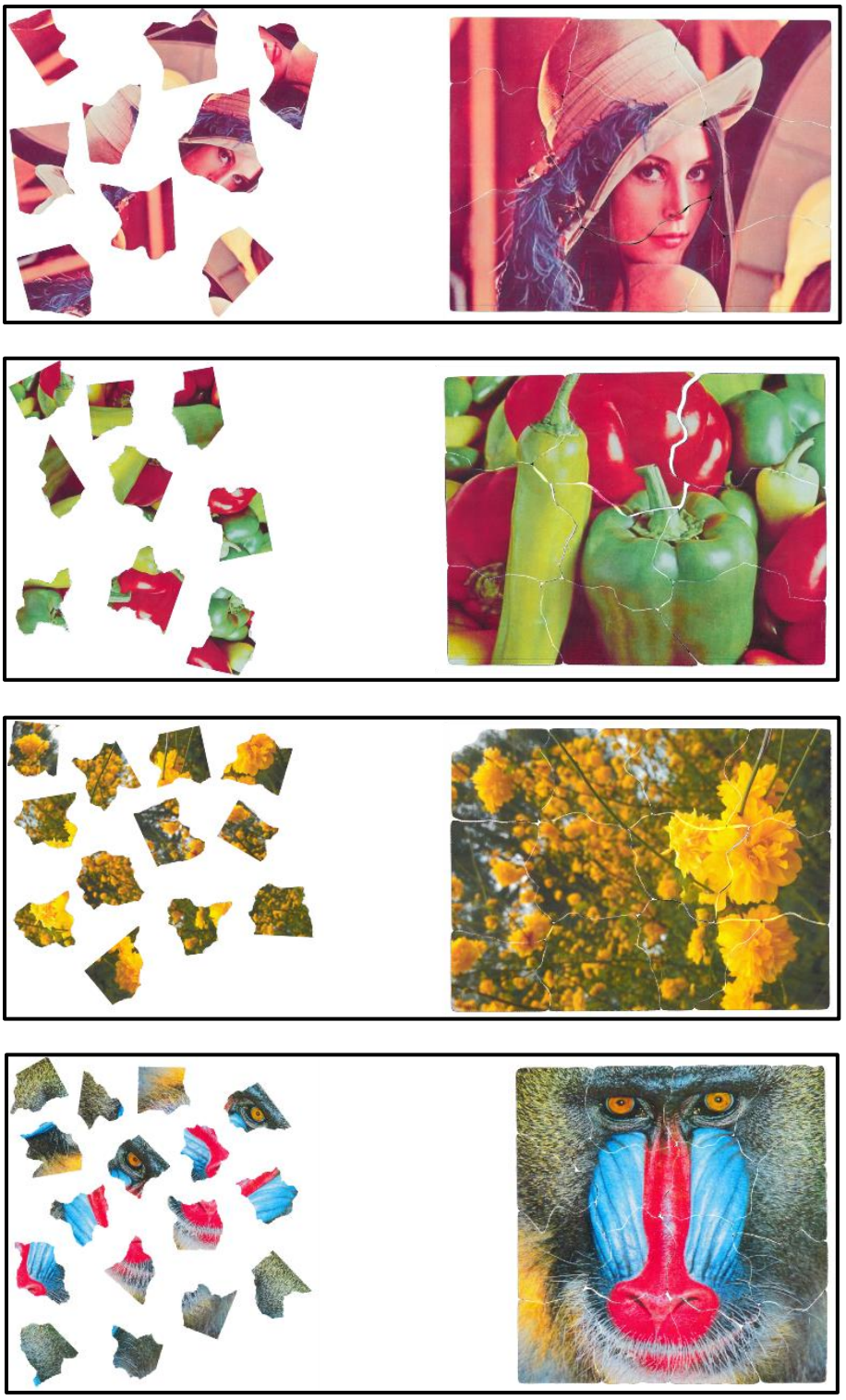


Figure 2.12: Demonstrating the effectiveness of out method on real hand-torn images. (a) 9-piece input data set, (b) Another 9-piece input data set, (c) 12-piece input data set, (d) 16-piece input data set.

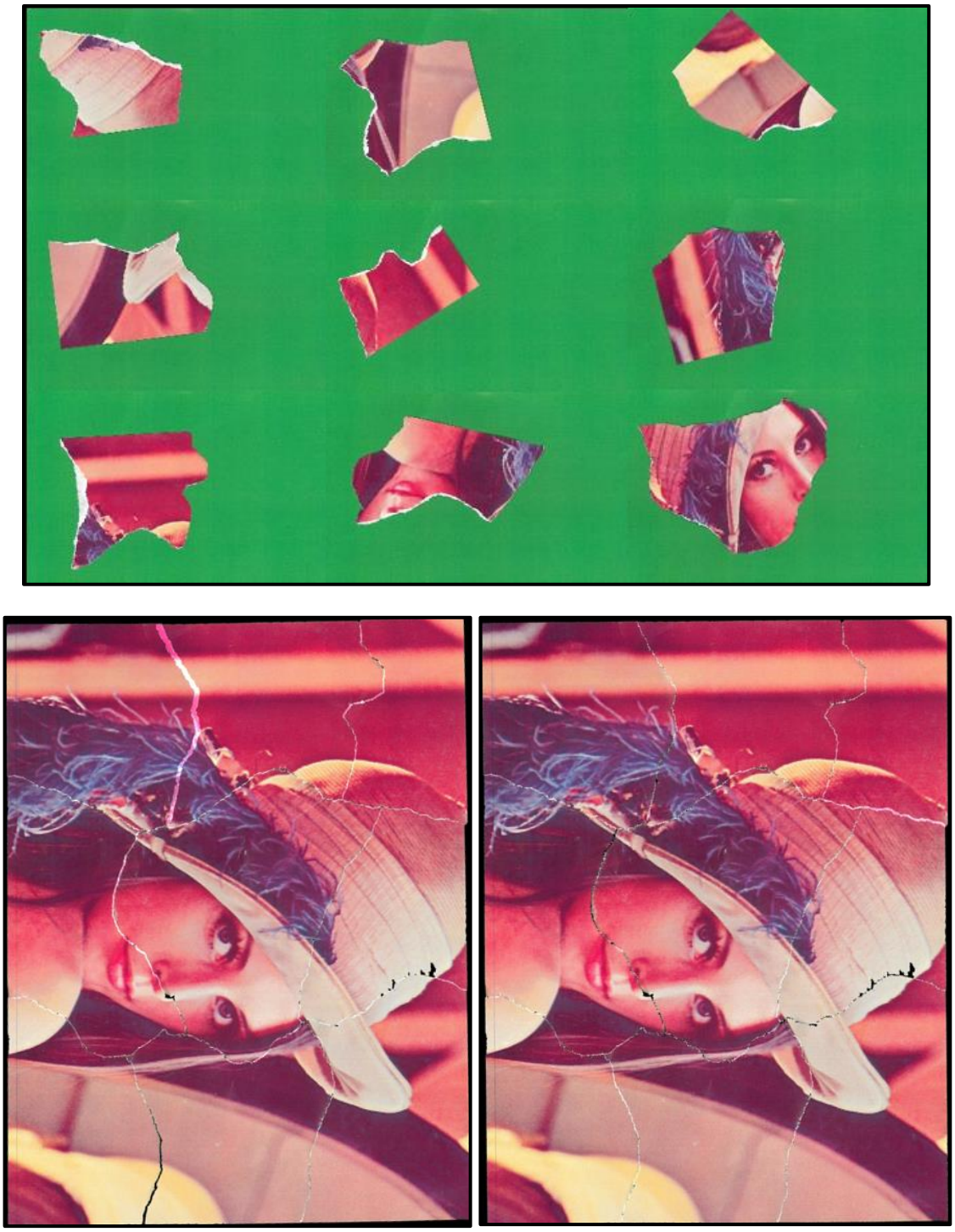


Figure 2.13: Comparing the results from our approach and others
(a) A data set of nine pieces before reassembly, (b) the result from our approach, (c) The result from an implementation of the approach in [6] (The two results are almost similar but our approach demonstrates lower time complexity)

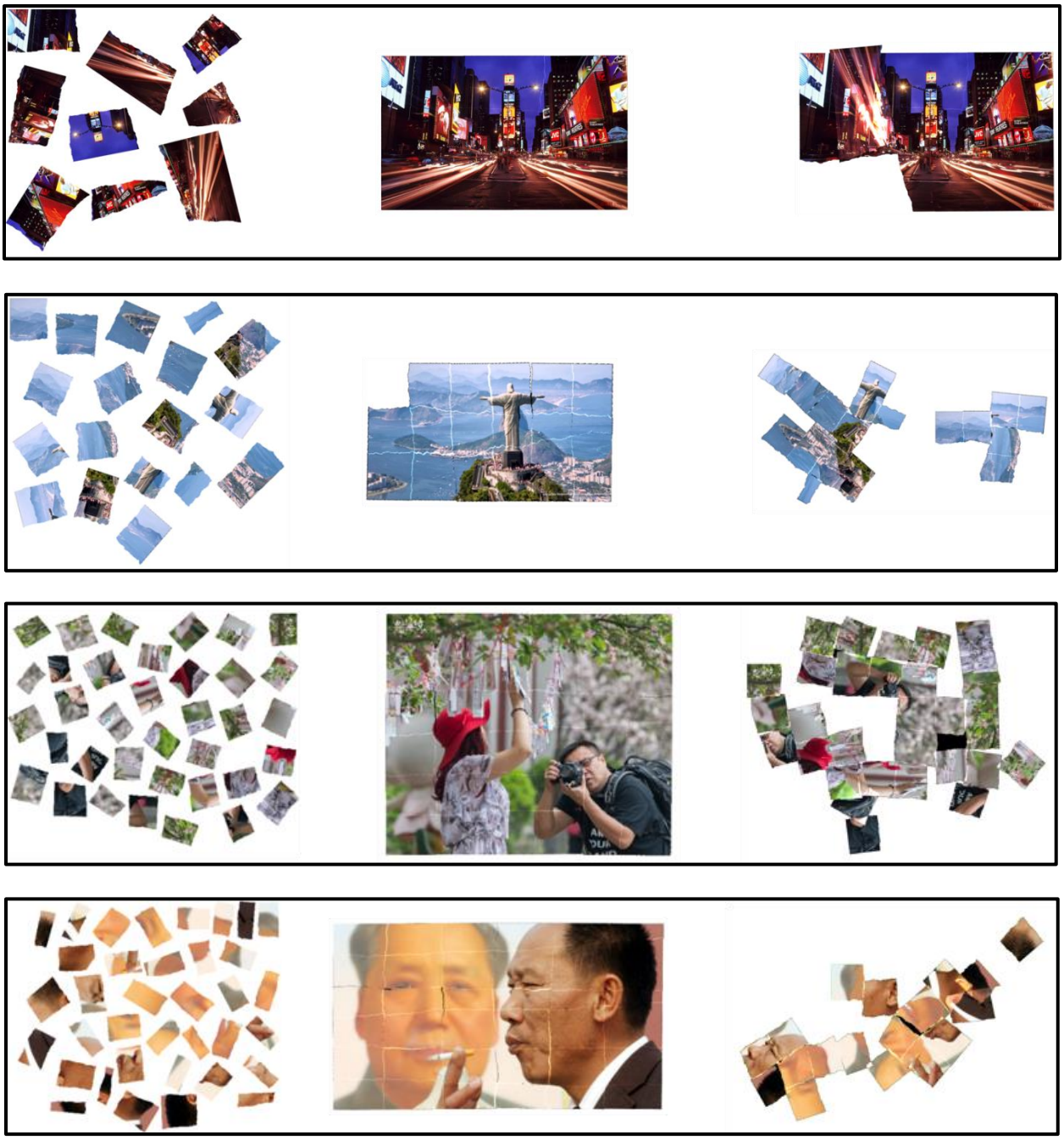


Figure 2.14: Comparing our method with the method proposed in [1]. First column is the input data set. Second column shows the result of our method and last column shows the result of [1]. (a) 9-piece input data set, (b) 18-piece input data set, (c) 32-piece input data set, (d) another 32-piece input data set.

Figure 2.14 shows the results. Most existing reassembly algorithms fail as the number of image fragments in the input data set increases. Figure 2.14 illustrates the capability of our approach when tested on a large input data set with 36 pieces for example.

2.4.1 Analysis and Comparison

When comparison with other methods using LCS, the pairings are determined based on how similar two segments are geometrically and color-wise. In order to decide whether a string of parings between segments is better than another or not, a variation of LCS approach is utilized. Others have used this method before but they have tried to pair pixels. In contrast to previously proposed methods, here we apply this method to curve segments based on their geometry and overall color properties, also how they are placed next to their neighbors. This makes our method more robust to possible noise on border pixels. Let us assume a total of R number of image fragments. Also, let us assume, for a random choice of a pair of image fragments, an average of N curve segments on first border and an average of M on second border, with an average length of S. Also, let us assume an average of U total pixels on the border of the first image and an average of V total pixels on the border of the second image. Therefore, it follows that $\mathrm{U} = \mathrm{N} \times \mathrm{S}$ and $\mathrm{V} = \mathrm{M} \times \mathrm{S}$.

In [6], in pairwise matching step of the algorithm, a scoring scheme that is based on revised version of ICP is introduced to rank potential matches between a pair of fragments. The pairwise matching algorithm runs for $\frac{\mathrm{R}(\mathrm{R}-1)}{2}$ times exactly for all possible pairs of image fragments which is of $\mathrm{O}(\mathrm{R}^2)$ complexity. During each iteration of pairwise matching, since the algorithm investigates the possibility of a potential match between each one of the curve segments on the first border and each on the curve segments on

the second border, $O(MN)$ potential matches are calculated between the pair of fragments at hand. For each one of potential matches ICP is run at least once to calculate the corresponding score. One iteration of ICP is of $O(UV)$ time complexity, since the algorithm investigates all the pixels on the whole border of the two fragments regardless of which pair of curve segments ICP is being run for. Therefore,

$$\text{total time complexity for pairwise matching step of } [6] = \quad (2.7)$$

$$O(R^2 \times M \times N \times U \times V).$$

Our proposed pairwise matching step runs an LCS-based algorithm on shape feature strings, shape feature being the descriptor calculated for each curve segment. Finally, a transformation based on the recovered matches is calculated. The pairwise matching step of our algorithm runs for all possible pairs of image fragments as well, which makes it of $O(R^2)$ complexity too. During each iteration of pairwise matching for two randomly picked fragments, the algorithms presented before to calculate the descriptors for curve segments and after that, the most similar sequence of curve segments between two fragment borders will each run exactly once after one another. Calculating a descriptor for a curve segment is of $O(S)$ since it investigates all pixels on the segment towards that purpose. Given that a descriptor is calculated for all curve segments on both borders, $O(N \times S) + O(M \times S)$, i.e. $O(U) + O(V)$, is spent to calculate the descriptors (lines 2—5). To calculate $\mathrm{Sim}(S_{i,k}, S_{j,l})$ for all $1 \leq k \leq n$ and $1 \leq l \leq m$, algorithm runs a nested loop of $O(MN)$ (lines 6—11) for which all the operations inside are of $O(1)$ (lines 8—11). Therefore, time complexity for the algorithm is $O(U) + O(V) + O(MN)$. Revised LCS will form a matrix, H, of size $M \times N$ (lines 6—8). The operations executed to calculate each element of H (line 8) is of $O(1)$, which makes time complexity for revised LCS of $O(MN)$.

Since all the algorithms mentioned above each run just once and one after the other, total time complexity to execute them will be of $O(U) + O(V) + O(MN) + O(MN)$ or simply $O(U) + O(V) + O(MN)$. Therefore, we can conclude that,

$$\text{total time complexity for our pairwise matching step} = O(R^2 \times [O(MN) + O(U) + O(V)]). \quad (2.8)$$

For low resolution image fragments all three terms in $O(U) + O(V) + O(MN)$ are almost of the same strength. As the resolution of the image fragments increase from case to case, the $O(U) + O(V)$ term dominates the other one, and the reason is that we hope to choose DP-threshold adaptively so that M and N do not vary too much with the varying resolutions of input data sets. In other words, we can replace $O(U) + O(V) + O(MN)$ with $O(U) + O(V)$ in equation 2.8 and be correct. We can also replace $O(U) + O(V)$ with $O(U)$ or $O(V)$ for obvious reasons; and still be correct. Therefore we can have

$$\text{total time complexity for our pairwise matching step} = O(R^2 \times U). \quad (2.9)$$

CHAPTER 3.
DOCUMENT REASSEMBLY

3.1 Introduction

In the final chapter, document reassembly is addressed and some ideas are discussed as future work. Document reassembly can be considered to be arguably the more complicated and more useful subcategory of image reassembly in general.

The problem of having to reconstruct shredded documents is often faced by historians and forensic investigators. Throughout history there have been many examples of a secret service destroying evidence by shredding them into pieces after they have been exposed. Solving this sort of puzzles manually is tedious and expensive due to the fact that the number of possible permutations of fragment arrangements increases exponentially with the number of constructing pieces. This is why it is necessary to have automatic or at least semi-automatic tools to help with the task of reassembly.

In document reassembly, depending on whether the pieces are hand-torn or machine shredded, we are dealing with a different type of problem with slightly different properties.

3.2 Related Work

The problem of reassembling shredded documents is closely related to the problem of reassembling an ordinary image. The difficulty of assembling a document arises due to the fact that in some cases the shredded pieces are too many in number or too small in size or have regular shape depending on the case. As a result, the geometric shape of fragments do not carry a lot of valuable information if the document has been machine-shredded rather than hand-torn. On the other hand, advantage in the reassembling of a document comes from the fact that matching partial words and context can be used as an additional tool to help improve the task of detecting pairwise matches between

potential neighboring pieces. Existing automatic document reassembly algorithms can generally be divided into multiple categories depending on whether they take advantage of this common property of document reassembly compared to plain image reassembly while trying to detect pairwise matches.

We approach the task of strip-shredded document reassembly as a graph optimization problem. As explained later in this chapter, the problem is converted into finding a Traveling Salesperson Path (TSP) with maximum weight in a weighted directed complete graph. The advantage of doing so is that TSP is a very well-studied problem in computer science and there already exist many different solutions, greedy of exact, to this problem. Here, the literature review is divided into two main categories. First is the literature that investigate document reassembly problem in general; and second is the literature that investigate that traveling salesperson problem.

3.2.1 Shredded Document Reassembly Problem in Literature

Richer et al. [15] try to solve the problem of document reassembly as a general image reassembly problem. They deal with hand-torn fragments and, using a SVM classifier, they identify locations on fragment pairs that belong together based on both shape and context local features. Therefore, the number of potential fragment matches will reduction while a sufficient number of correct matches are preserved. Finally an iterative algorithm is used to align groups of matching fragments until the document has been restored. Justino et al. in [4] have a similar approach towards document reassembly problem. They use a polygon approximation to reduce the complexity of the fragment boundaries and then extract relevant geometric features of the polygons, such as angles on the vertices and the length of line segments on estimated polygons, in order to perform local

reconstruction and detect pairwise matches. The ambiguities resulting from the local reconstruction are resolved and the pieces are merged together as a result of a search for a global solution. Marques and Freitas deal with strip-shredded-documents in [16] and they also try to solve the reassembly problem by feature matching. They use color features of pixels on the borders of the fragments as the only type of feature. Their baseline system has three stages. The first stage is the feature extraction from the boundary based on color pixels using two different color models: HSV and RGB. In the second stage the Nearest Neighbor Algorithm is used as the base for calculating the Euclidian distance between the feature vectors of the distinct strips. The third stage is to compare the distances and apply the method called winner-takes-all indicating, in this case, that the shorter distance is the more likely that those strips will fit together. In all [15] and [4] and [16], through the process of finding pairs of potential neighboring fragments, no attempt is made to try to use semantic features such as matching partial words which is a common feature of all documents. It is due to this observation that these papers and papers approaching the problem with similar ideas are in the same category when it comes to solving the problem of document reassembly.

On October 27 of 2011 DARPA presented public with a shredder challenge. In order to evaluate the possibility of reconstructing shredded documents, a test set of five puzzles of increasing difficulty was created and presented to public. Some of these puzzles have pieces exceeding multiple thousands. Deever and Gallagher [17] were successful in solving two of the presented puzzles completely by proposing a semi-automatic approach. Automatic algorithms are proposed for segmenting individual shreds. Using principle component analysis, all extracted pieces are oriented vertically and through

some shape analysis upside is detected for all pieces in order to form a consistent set of input fragments. Next, features are computed based on the location of ruler lines or where the marks meet the border of a piece and are used for ranking potential matches for each shred. At this point, the size of the problem has reduced considerably and it can be handled manually through human-computer interaction. Shang et al. [18] propose a semi-automatic reassembly system to solve the puzzles presented by the DARPA challenge as well. Through a curve matching scheme, they make a list of most probable neighboring pieces for each shredded fragment. Afterwards, using user interaction, they narrow down the choices and try to reassemble the original document by adding correct matching pieces to an already partially reassembled document. The novelty of their method lies in the curve matching step where they allow overlap between two pieces that are being investigated for a pairwise match. Allowing overlapping of pieces during matching helps cope with shape deformations due to shredding. Both [17] and [18] can be considered belonging to the same category of document reassembly literature due to the fact that the alignment of text lines, crossing characters and color information on the pieces, as properties exclusively belonging to documents rather than images, are utilized to improve pairwise matching performance. On the other hand, the groupwise matching step in this group of literature is usually handled through human-computer interaction due to the large number of pieces in the puzzle, making these methods automatic only in part.

Reassembly of fragment objects from a collection of randomly mixed fragments is a common problem in classical forensics. Shanmugasundaram et al. address the digital forensic equivalent of this problem in [19]. They solve the problem of reassembling a digital file that has been divided into pieces and stored on different locations on storage

medium. To solve this, a candidate probability for adjacency between each pair of fragments is calculated using a context-based statistical model that has been trained over all available pieces. The problem of finding the optimal permutation of fragments among all possible permutations is formulated as finding a maximum weight Hamiltonian path in a complete graph.

3.2.2 Traveling Salesperson Problem in Literature

Traveling salesperson problem (TSP) is a graph optimization problem. Graph G is a directed weighted graph and TSP is the problem of finding a path the goes through each vertex of the graph once and only once such that the summation of weights of the edges in the path is minimized. Due to being applicable to many daily optimization problems, TSP has been the subject of extensive research and studies since 1950s in different scientific circles. Early attempts at solving the problem were only successful for graphs with small number of vertices in the order of 10s. It was not until 1960s when a new approach to the problem began to emerge that instead of seeking optimal solutions one would look for a near optimal solution whose length is provably close enough to the optimal length. Karp's publication in 1972 [21], showing TSP to be NP-complete, was a good mathematical explanation for this turn towards heuristic method. It is the result of this observation, that approaches to solve TSP problem can be grouped into two main categorizes, the first one being the heuristic methods and the second one being algorithms that solve the problem for the one and only one optimal solution, i.e. exact algorithms.

Building the tree of all Hamiltonian paths starting at a specific vertex in a graph can be represented as a tree. The optimal Hamiltonian path, i.e. TSP only optimal solution, is

one of the branches in this graph. However, the tree expands exponentially as the length of the paths increase, making it impossible to examine each path in the tree seeking the optimal solution. Therefore, heuristic methods are proposed to solve this problem. Often call greedy heuristics, in these approaches, each step looks good but it does not look ahead to make a decision. One efficient searching strategy frequently used in literature to search for a near optimal branch in other combinatorial problems or TSP specifically is $\alpha\beta$ pruning method. The final revision of this algorithm has been proposed by D. Knuth and R. Moore [31]. It is a search algorithm that seeks to decrease the number of nodes that are evaluated in the search tree. It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. In other words, it prunes away branches that cannot possibly influence the final decision.

Based on a concept Clarke and Wright [22] presented, Golden [23] proposed a heuristic method with time complexity of $O(n^2 \lg(n))$ which worst case ratio is bounded by a linear function of $\lg(n)$. Another approximation approach is called insertion method, also known as interpolation method, presented by Rosenkrantz et.al. [24]. An insertion procedure takes a sub-tour of k nodes at iteration k and attempts to determine which node not in the sub-tour should join the sub-tour next (the selection step) and then determines where in the sub-tour it should be inserted (the insertion step). There are multiple approaches that can be taken in both selection step and insertion step which will affect the performance of the algorithm. However, it is safe to say that most of these approaches perform in $O(n^2)$. When it comes to worst-case ratio, the best-known algorithm is presented by Christofides [25]. The solution to his algorithm, given in 1976, at worst is 1.5 times longer

than the optimal solution. Christofides algorithm, at one of its steps, involves finding a minimum-weight perfect matching for a subgraph, which is a process of $O(n^3)$ and happens to be the bottleneck for his algorithm. Therefore, the running time for the whole algorithm will end up being of $O(n^3)$ as well. The drawback for Christofides approach is that it is applicable only to graphs whose edge weights for a metric space (they are symmetric and obey the triangle inequality) and, therefore, is not useful for directed graphs. Kim's [26] method can be viewed as a modification of Christofides algorithm and also requires a metric space for edge weights. It has an improved running time of $O(n^2)$, but has a higher worst-case ratio of 2 when compared with Christofides method. Karp [21] presents a partitioning algorithm for the TSP in the plane and performs a probabilistic analysis in order to obtain results. From a practical point of view, his procedure is a decomposition algorithm, which is capable of solving extremely large TSPs. The key idea is to partition a large rectangle into a number of sub-rectangles and solve a TSP in each sub-rectangle. The outcome is an Euler cycle, which can be transformed, into a tour over all the nodes with minimal effort.

All the heuristics mentioned above have one property in common. In all of them, a solution is being built from scratch starting with nothing. Another approach to finding a greedy solution to the TSP problem is improvement methods. These methods start with a feasible solution and look for an improved solution that can be found by making a small number of changes to the solution currently at hand. The best-known example for this category is presented by Lin [27] in 1969, also known as branch exchange algorithm. The branch exchange procedure terminates at a local optimum. For a given k, if we define a k-change of a tour as consisting of the deletion of k edges in a tour and their replacement by k other

edges to form a new tour, then a tour is k-optimal if it is not possible to improve the tour via a k-change. These branch exchange procedures are important since they illustrate a general approach to heuristics for combinatorial optimization problems. In addition, they have been used to generate excellent solutions to large-scale traveling salesman problems in a reasonable amount of time. This method starts with a feasible solution and tries to reach a k-optimal tour through repeated k-change operations. The running time for this method is of $O(n^k)$.

Finally, a third approach to heuristic solution of a TSP problem is composite heuristics [28]. A typical composite procedure would obtain an initial tour using one of the tour construction procedures as the first step. The second step would be to apply an improvement procedure, like a k-optimal, to the tour found as the result of first step.

Besides the heuristics approaches, the other category to solve TSP algorithm is exact algorithms. The exact algorithms seek the only optimal solution to the problem and just a near optimal one is not acceptable. The drawback with these methods is that they are only applicable to small problem sizes, i.e. graphs with small number of vertices. Because as the number of vertices increase in a graph, the search space grows exponentially and makes it impossible to seek the exact optimal solution. Dantzig et.al, in what is considered the seminal paper on the subject [29], express the problem as an integer linear program formed by two constraints in the model and then develop the cutting plane method by adding inequality constraint to gradually converge at an optimal solution. Generally, when people apply this method to find a cutting plane, they often depend on experience. Therefore, this method is seldom deemed as a general method. Little et.al. [30] use branch and bound method to find the solution. Branch-bound algorithm is a search

algorithm widely used, although it is not good for solving the large-scale problem. It controls the searching process through effective restrictive boundary so that it can search for the optimal solution branch from the space state tree to find an optimal solution as soon as possible. The key point of this algorithm is the choice of the restrictive boundary. Different restrictive boundaries may form different branch-bound algorithms. The Held–Karp algorithm, also called Bellman–Held–Karp algorithm, is another algorithm proposed in 1962 independently by Bellman and by Held and Karp to solve the TPS based on dynamic programming. This algorithm runs in $O(n^2\ 2^n)$. Improving this time bound seems to be difficult. For example, it has not been determined whether an exact algorithm for TSP that runs in time $O(1.9999^n)$ exists.

3.3 Methodology

Although prior research in the related fields of jigsaw puzzle solving and reconstruction of hand-torn documents provide insights on how to approach the problem, reconstruction of shredded documents has its own unique difficulties. The first difficulty arises due to the size and shape of the pieces. Shredders are designed to make pieces that are as small as possible making features that depend on color, texture and character crossings more error prone. Further, shreds are of similar shapes, making shape a less distinctive feature. Another difficulty is the deformations introduced at the edges of the shreds due to the shredding process, making accurate pairwise matching very difficult. This problem is further exacerbated by the complications related to scanning, which may involve loss of some pieces and variations in color leading to incorrect segmentation of individual piece images. Another difficulty concerns the number of shredded pieces that need to be considered during reconstruction. Overall, it can be stated that shredded document

reconstruction is a more complex problem than the ones addressed in related research fields.

On the other hand, there are some features that are exclusively related to document fragments. For example, when dealing with documents, it is common to come across words that are cut in half due to the process of fragmentation and are straddled on the two opposing sides of a cut line on the border of two neighboring pieces. Text lines can be mentioned as another example which are present in some documents to guide the direction of writing. This common features among document pieces can be used as a tool to detect accurate pairwise matchings between pair of potentially neighboring pieces in addition to the previously used color features and geometry features which may prove not to be sufficient when talking about document reassembly.

3.3.1 Pairwise Matching

While trying to detect a possible pairwise match between a pair of fragment pieces, first the pieces need to be segmented and their border need to be extracted. One way of finding the best match between two pieces is to try all possible offsets while aligning the pieces along their border segments that they potentially share. For puzzles with a large number of shreds this can prove to be intractable and unnecessary. The proposed method is to classify border pixels into background, text lines and marks. It is known that true matches will have text lines that extend across the border. Further, it is expected that text marks near the edge of one shred will often extend across the boundary and into the neighboring shred. Therefore, instead of having to try all possible offsets along the potentially shared border segment of a pair of pieces, only offsets will be tried that align a text line in one with a text line in the other. Other color and geometry features can also

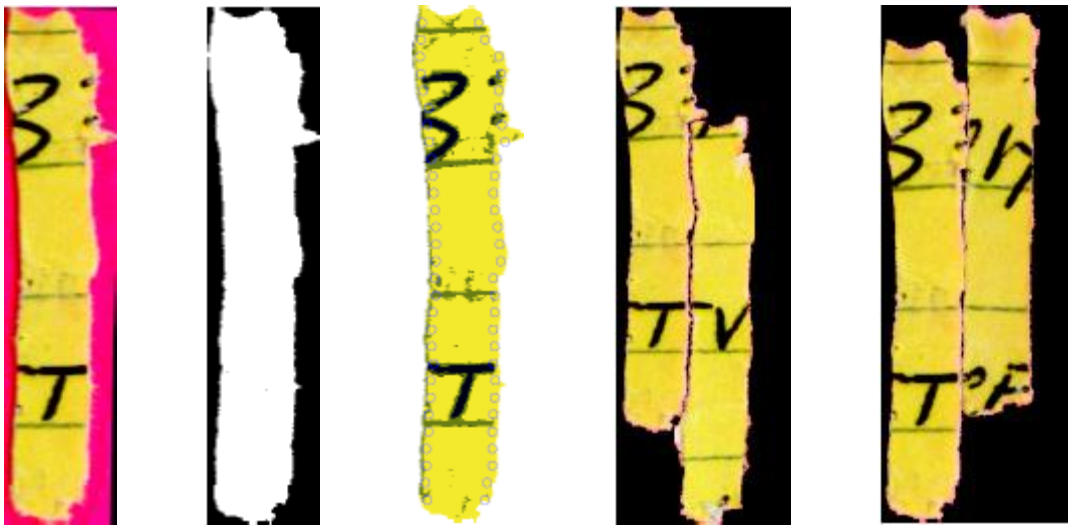


Figure 3.1: Document reassembly pairwise matching
(a),(b) Shredded piece segmented and its border extracted, (c) border pixels are sampled and classified, (d) a valid potential match after one possible alignment of text lines, (e) another valid potential match after another possible alignment of text lines.

be used to verify the validity of a potential match. Figure 3.1 is illustrating this concept. After determining a set of valid potential matches between a pair of pieces, next step is to assign probabilities to the likelihood that a valid match is in fact the correct match between that specific pair of pieces. In a valid potential match, the text line from one piece extends across the border into the neighboring piece. Therefore, to calculate the probability value for a valid match, we are proposing to use optical character recognition algorithms to detect characters close to the border of each piece. One easy way to use this information is for these characters to be matched against one another to see whether they form a valid word in the dictionary of the underlying language of the document. Another possibility is to develop a statistical model that helps us predict the occurrence probabilities of a group of symbols together. Figure 3.2 depicts the general idea being proposed here. Tesseract can be a viable option as an optical character recognition algorithm. It has been considered as one of the most accurate open-source OCR engines

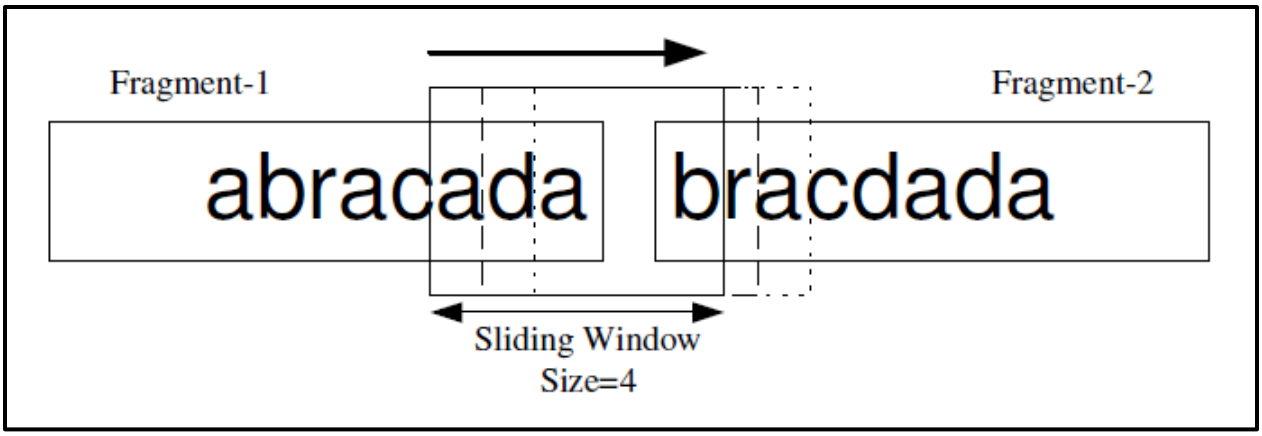
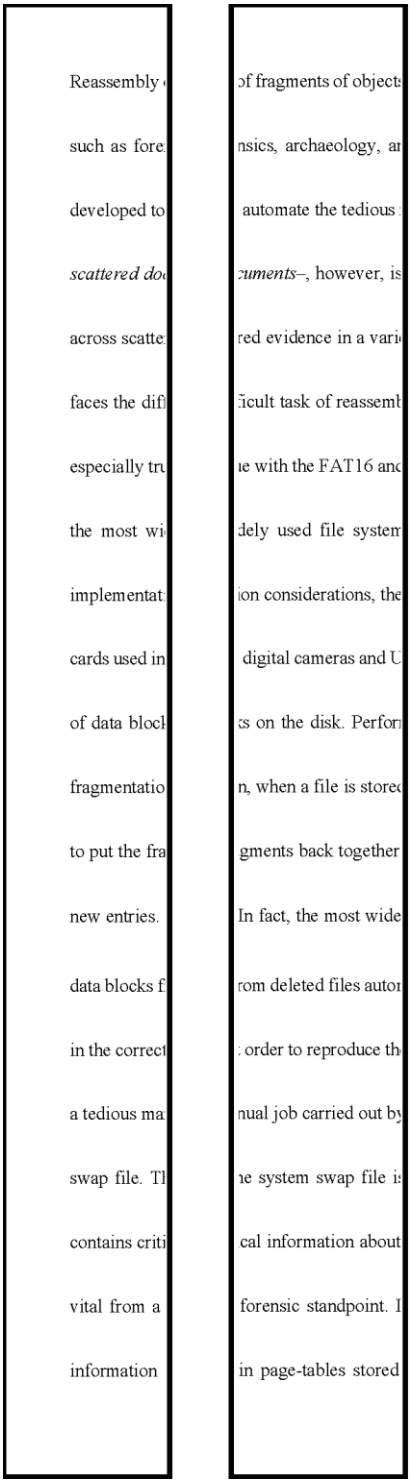


Figure 3.2: OCR near the borders of potentially matching pair

that has been developed for various operating systems under google sponsorship since 2006 [63].

In this document, we tackle the problem of reassembling strip-shredded document reassembly. For a strip-shredded document, it can be assumed, with high probability, that the lines of text are already aligned. Therefore, the half-word at the right border of the first shred and the half-word at the left border of the second shred are extracted after OCR and are given a probability that represents the chance of them forming a valid word when put together. This is done by trying to match the composed word with all the words present in a dictionary. In the composed word, the letters close to the border area are likely to have been damaged or recognized incorrectly during the OCR process. It is not wise to rely on such characters while trying to find matches with dictionary words. For this reason, we deploy Longest Common Subsequence method to find matches between the characters of our composed word and the words in the dictionary. LCS allows for deletion of a character if that helps finding a better match between the characters of the two words that are being compared. The largest score that LCS calculates as a result of comparing our composed word with all the words in the dictionary can be considered to be the



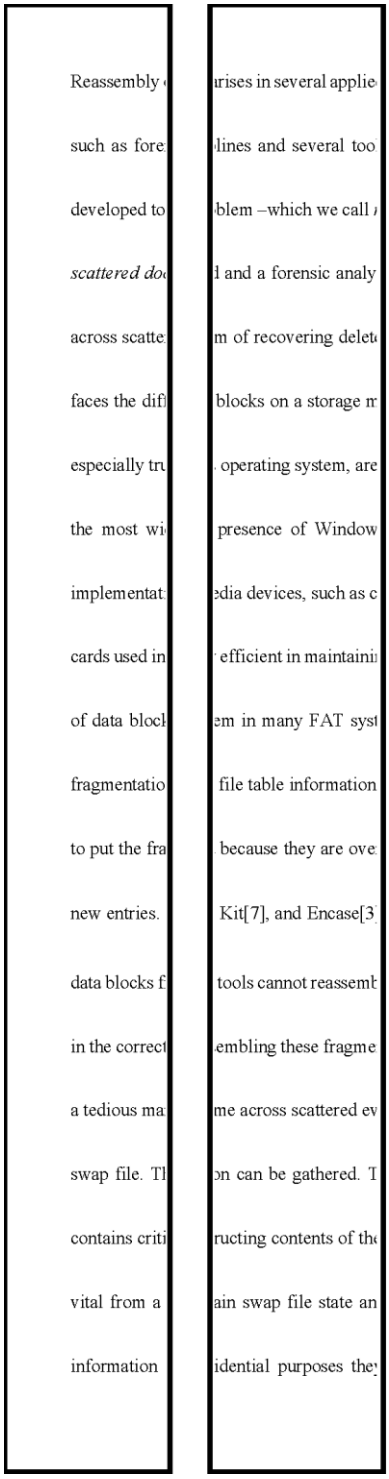


Figure 3.3: Neighboring pair of shreds vs. a non-neighbor pair
(a) The probability of this pair being neighbors is calculated to be 0.85, (b) the probability of this pair being neighbors is calculated to be 0.61.

likelihood of that composed word to be a valid word, after going through a normalization process of course. Figure 3.3a depicts a pair of neighboring shreds and figure 3.3b depicts a pair of shreds that are in fact not neighbors.

For each pair of shreds, we calculate a score that represents the likelihood of that pair being neighbors. This score is calculated based on the probability of the composed wordson the common border of the two shreds. This score for the pair in figure 3.3a is 0.85 and for the pair in figure 3.3b is 0.61. As expected the neighboring relationship can be detected based on this score correctly.

3.3.2 Groupwise Matching

Assuming we are dealing with strip-shredded documents (in strip-shredded documents, the strip runs the length of the document), once the adjacency probabilities are assigned for all pairs of pieces, the permutation of the fragments that leads to correct reassembly, among all possible permutations, is likely to maximize the sum of probabilities of adjacent fragments. This observation gives us a technique to identify the correct reassembly with high probability. This permutation is most likely to be the one that leads to correct reconstruction of the document. The problem of finding a permutation that maximizes the sum of probabilities of adjacent fragments can also be abstracted as a graph problem. To do this we take the set of all probabilities to form an adjacency matrix of a complete weighted graph of n vertices, where vertex i represents fragment i and the edge weights quantify the probability of two corresponding fragments being adjacent. The proper sequence is a path in this graph that traverses all the nodes and maximizes the sum of probabilities along that path. The problem of finding this path is equivalent to finding a maximum weight Hamiltonian path in a complete graph. Figure 3.4 is showing a simple

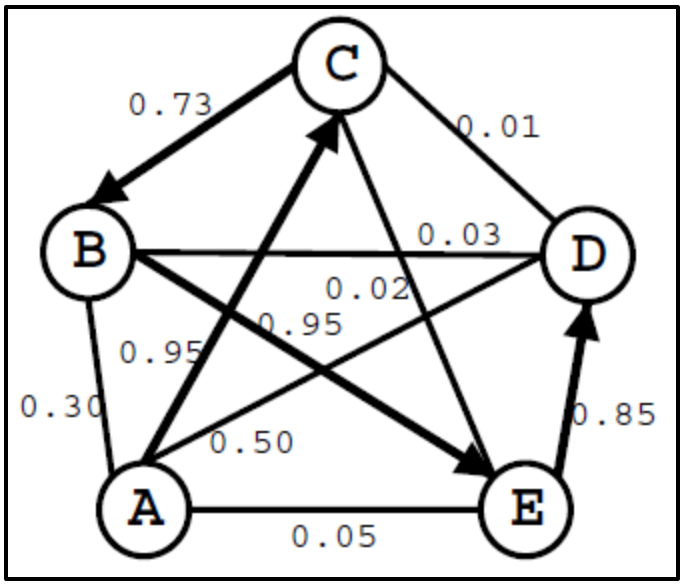


Figure 3.4: Groupwise matching for document reassembly

example to describe this idea. The optimum solution to this problem turns out to be intractable. However, there are many heuristics known in the literature, as discussed in literature review section, and we employ one such heuristic to solve this problem. The heuristic that we have deployed to solve this TSP problem is called αβ pruning. Even though αβ pruning is not an exact algorithm and only a greedy method that grantees a near optimal solution to the TSP, since the pairwise matching scores calculated in the previous step are somewhat trustworthy, this search algorithm suffices in our case. A pseudocode for this search algorithm is presented in figure 3.5.

Using this search algorithm, a path is found that goes through all of the vertices and the summation of the edges along the path is maximized. This means that this path specifies an ordered list of vertices, i.e. a permutation of the shredded document fragments for which the consistency of shredded pieces as a group is maximized. As long as the number of shredded pieces in our problem is not too big and the pairwise matching scores are reliable, we can trust this algorithms to return the optimal solution. As the number of

```
// 1) there is no beta in this pseudocode. We assume beta = 1. The depth
// for search is alpha = a.
// 2) n is the number of fragments
// 3) vertex is a class whose objects represent image fragments
// 4) node1 represents the first fragment in the reassembled image
// 5) path (a global variable) is an ordered list of vertices and will
// eventually hold the final answer
// 6) alpha(node, a) returns a scalar value, the maximum cost among all
// possible paths of length a that originate at node.father and pass
// through node
// 7) node is an object of class vertex
// each node has a pointer to its father (another object of type vertex)
// each node has a pointer to a vec. of vertices which keeps its children
// (it consists of any vertex that is not in the global variable path yet)
// each node has a pointer to a vertex which is the child that comes after
// it in the optimal path (final answer)
// 8) c(node.father, node)is the weight on the edge (on the complete
// weighted directed graph) that goes from a node's father to that node
01 node := node1
02 path.append(node)
03 for n-1 times
04    v := -∞
05    for each node.children
06       Temp := alpha(node.children[i], a)
07       if (temp > v)
08          v := temp
09          node.bestChild := node.children[i]
10    node := node.bestChild
11    path.append(node)



01 function alpha(node, length)
02    if length = 1 or node is a terminal node
04       return c(node.father, node)
05
06    v := -∞
07    for each node.children
08       Temp := c(node.father, node) + alpha(node.children[i],length–1)
09       if (temp > v)
10          v := temp
11    return v
```

Figure 3.5: Pseudocode for αβ pruning

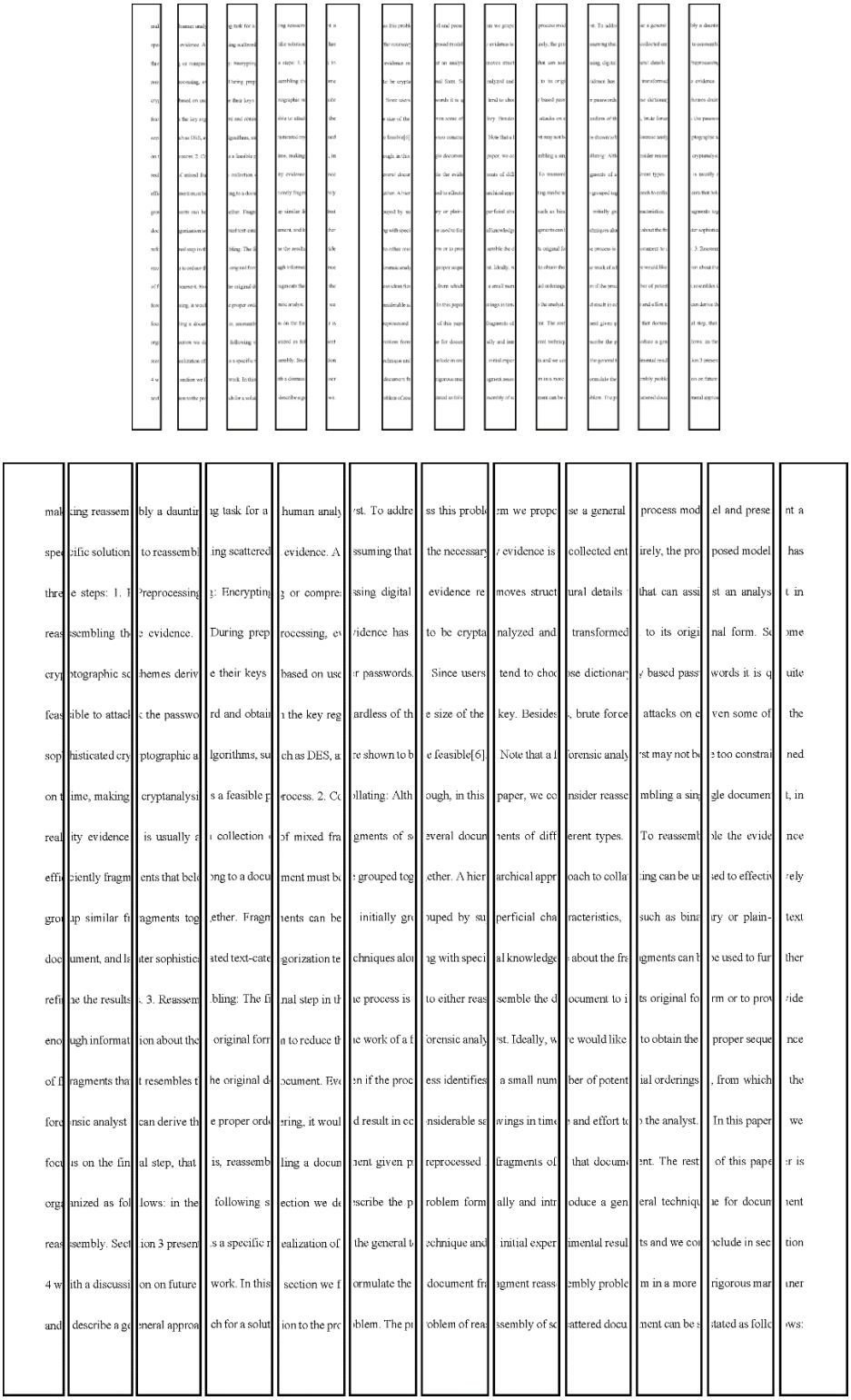


Figure 3.6: result of the shredded document reassembly
(a) before the reassembly, shreds are out of order, (b) final result of the reassembly.

shreds increase, or the pairwise matching calculation gets more complicated, this algorithm is guaranteed to only return a near optimal solution. In such a case, this solution has to be viewed as a preliminary giant step toward a manual reassembly of a shredded document. The other option will be to use an exact algorithm, like the ones mentioned in the literature review section, toward solving the TSP problem and recover the one and only one optimal solution to our problem.

3.4 Experimental Results

In this section, we present the result of the reassembly of a document that has been shredded into 12 pieces. The shredding process is simulated. Google has been used for the optical character recognition purposes. Figure 3.6a shows the shreds before finding the correct order and figure 3.6b shows the result of the reassembly.

Figure 3.7 shows the result of reassembling an 18-piece document using our method. In this specific case α is set to be equal to 6. The shreds are grouped into 3 categories. The order in which shreds appear, in each one of these categories, are correct when considered separately. But order of categories themselves is not correct and they need a slight reshuffling.

Figure 3.8 shows the result of reassembling the same document, i.e. the 18-piece document. However, this time, instead of using $\alpha\beta$ pruning for groupwise matching step to solve the traveling salesperson problem, greedy approach has been deployed to solve the TSP problem. As a result, the shreds are grouped into categories with smaller number of shreds compared with figure 3.7. Not only groups with correct order of shreds are smaller in size when compared to previous method, but also the categories need more intense reshuffling to result in a completely correct reassembly of the document compared

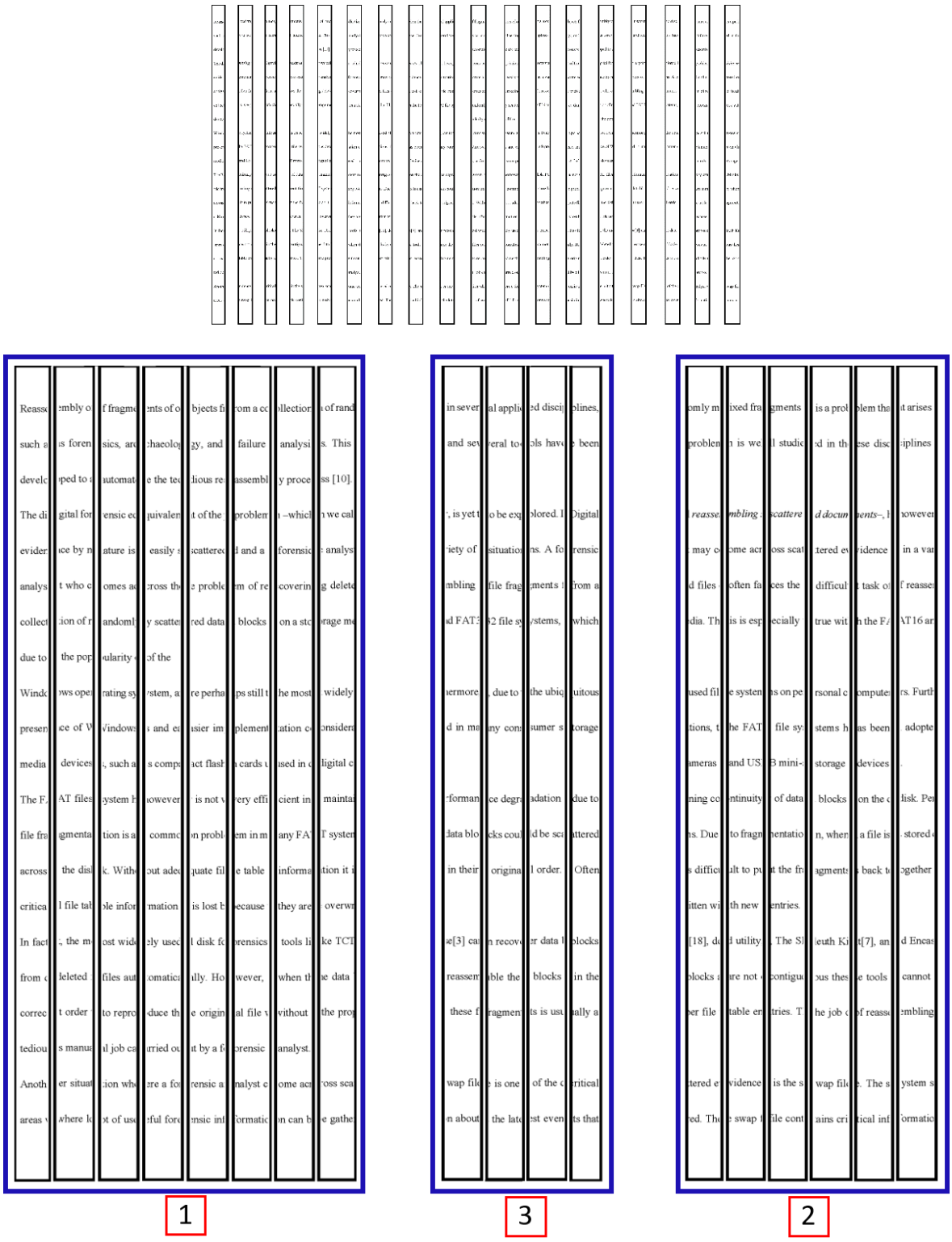


Figure 3.7: result of the shredded document reassembly
(a) before the reassembly, shreds are out of order, (b) final result of the reassembly when αβ pruning is used with $\alpha = 6$.

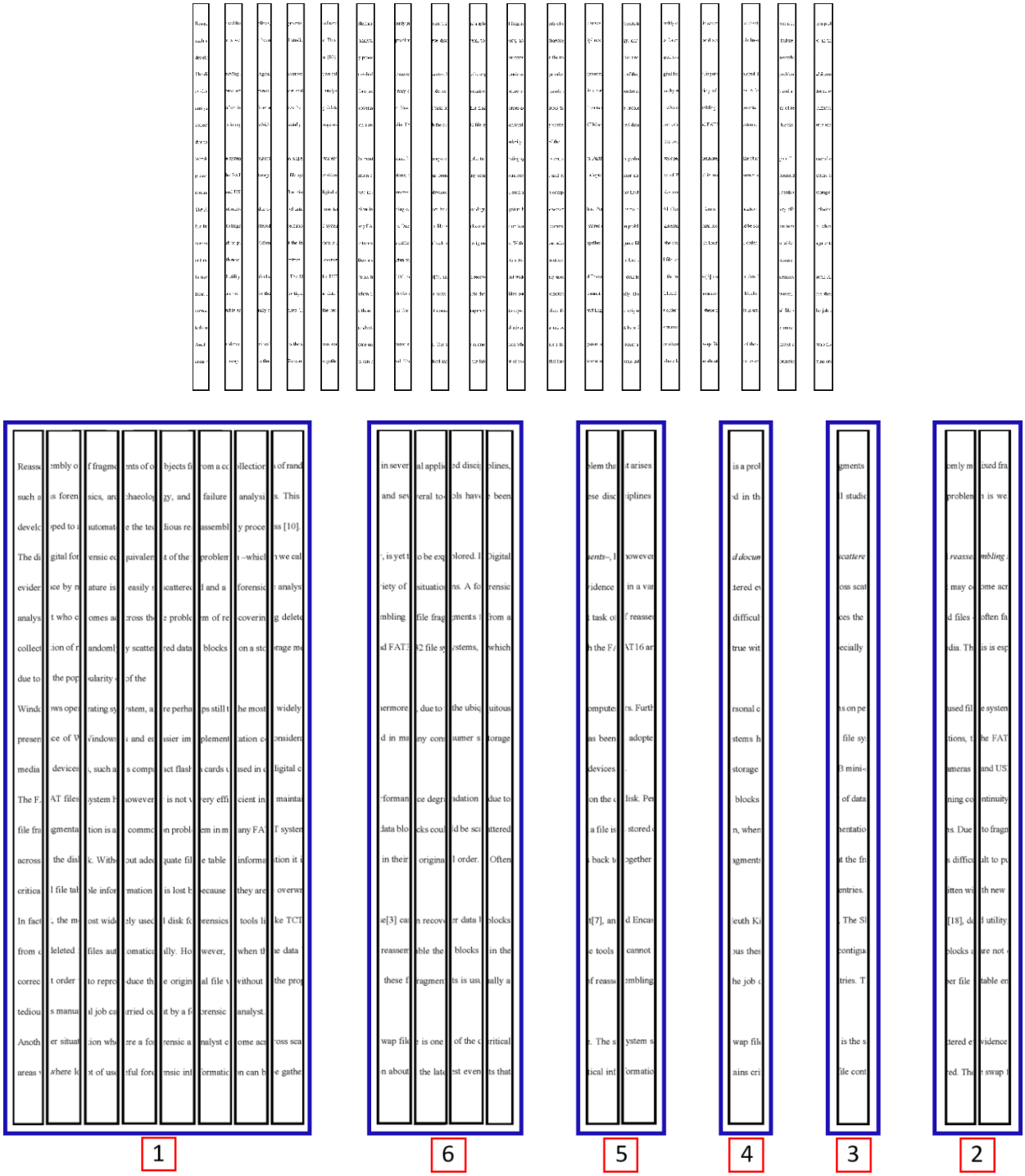


Figure 3.8: result of the shredded document reassembly
(a) before the reassembly, shreds are out of order, (b) final result of the reassembly when greedy method is used, i.e. $\alpha\beta$ pruning with $\alpha = 1$.

to when αβ was used. This example serves to show that deploying αβ is advantageous compared to simply using greedy method for groupwise matching.

The other option for solving the TSP, apart from greedy method and αβ pruning, is exhaustive search. This method has been investigated through an example with 12 pieces. The search space is so big in this case that causes the running time to grow exponentially and makes this method impractical in reality.

CHAPTER 4.
CONCLUSION

In this work, we investigated two main applications of image processing that are palpable in daily life. First, we introduced a novel idea, frequency division multiplexed imaging, to create a new imaging system, which will help us in different applications in the field of computational photography. Second, we dealt with the problem of 2D fragmented image reassembly and strip-shredded document reassembly.

One possible application of frequency division multiplexed imaging is stereoscopic image capture. Two images were captured simultaneously through the use of a beam splitter, two gratings, and a novel optical setup. Through proper optical arrangement, stereoscopic view of a scene can be obtained. Similarly, camera arrays where multiple images need to be captured can be implemented with a single camera and through spatial light modulation to place different images on different parts of the Fourier domain.

With dynamic spatial light modulation, it is possible to capture sub-exposure images. This can be used to implement a high-speed imaging system with a standard camera as the gratings can be changed very fast. Being able to capture sub-exposure images can be used in motion deblurring, object segmentation based on motion, high-dynamic-range imaging, and focus stacking as well. Optical modulation and Fourier domain allocation may also lead to new applications in compressed sensing.

In 2D fragmented image reassembly problem, we present a novel computational pipeline for the automatic reassembly of fragmented images. It consists of three main steps: removing artifacts, pairwise matching between two image fragments, and global fragment reassembly. We evaluate our algorithm using various real-world images and demonstrate that it is fast and robust. We deal with artifact on the border area by removing them

automatically. Our method to approximate the border curve of each image fragment is based on DP algorithm. But through smoothening the boundary curve and employing the curvature values of the pixels on the border, we proposed a revised DP algorithm which is novel and creates better and more robust polygonization results. Eventually, formulating the problem of finding the longest common subsequence of polygon sides shared by two potentially neighboring border polygons into the problem of finding the longest common subsequence shared by two strings in another contribution of our work. On the global matching front, more effective graph searching and backtracking algorithms or some stochastic optimization strategies could be investigated. In addition, adaptive algorithm to pick the proper DP-threshold for different data sets with different image resolutions can help generalize our algorithm even more.

The novelty of our method, when dealing with the problem of reassembling strip-shredded documents, lies in the fact that we formulated the global matching step of our algorithm as a famous graph optimization problem, i.e. traveling salesperson problem. TSP is a very well-studied problem in the field of computer science. We adopt a heuristic method to recover a near optimal solution to our TSP. For pairwise matching step, the method we use to calculate the neighborhood probability between two strip shreds is another contribution of our work. However, there is still much room for improvement when it comes to proposed method of calculating neighborhood probability between a pair of potentially neighboring strip-shreds. The reliability of the OCR result when it comes to the letters on the cut is questionable. Through this observation, in a more effective method to compare concatenated word with the words in the dictionary, the letters that are on the cut need to

be treated differently from the other letters that we have more confidence in. By devising a more sophisticated score calculating algorithm, the results can improve.

REFERENCES

[1] E. Tsamoura and I. Pitas. Automatic color based reassembly of fragmented images and paintings. *IEEE Transactions on Image Processing*, 19(3):680–690, Mar. 2010.

[2] S.H. Oguz, Y.H. Hu, and T.Q. Nguyen. Image coding ringing artifact reduction using morphological post-filtering. In *IEEE 1998: Proceedings of the Second Workshop on Multimedia Signal Processing*, pages 628–633, Dec. 1998.

[3] H.C. da Gama Leitao and J. Stolfi. A multiscale method for the reassembly of two-dimensional fragmented objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1239–1251, Sep. 2002.

[4] E. Justino, L.S. Oliveira, and C. Freitas. Reconstructing shredded documents through feature matching. *Forensic Science International*, 160(2–3):140–147, 2006.

[5] L. Zhu, Z. Zhou, and D. Hu. Globally consistent reconstruction of ripped-up documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):1–13, Jan. 2008.

[6] K. Zhang and X. Li. A graph-based optimization algorithm for fragmented image reassembly. *Graphical Models*, 76(5):484–495, 2014.

[7] P.J. Besl and N.D. McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb. 1992.

[8] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Molecular Biology*, 147(1):195–197, 1981.

[9] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, Dec. 1973.

[10] L. Cinque, G. Ciocca, S. Levialdi, A. Pellicano, and R. Schettini. Color-based image retrieval using spatial-chromatic histograms. *Image and Vision Computing*, 19(13):979–986, Nov. 2001.

[11] W. Kong and B.B. Kimia. On solving 2D and 3D puzzles using curve matching. In *IEEE 2001: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II-583–II-590 vol.2, 2001.

[12] F. Amigoni, S. Gazzani, and S. Podico. A method for reassembling fragments in image reconstruction. In *IEEE 2003: Proceedings of 2003 ICIP*, pages III-581–4, Sep. 2003.

[13] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[14] H.J. Wolfson. On curve matching. *IEEE Transaction on Pattern Analysis and. Machine Intelligence*, 12(5):483–489, May 1990.

[15] F. Richter, C.X. Ries, N. Cebron, and R. Lienhart. Learning to reassemble shredded documents. *IEEE Transaction on Multimedia*, 15(3):582–593, Apr. 2013.

[16] M.A.O. Marques, and C.O.A. Freitas. Reconstructing strip-shredded documents using color as feature matching. *Proceedings of the ACM Symposium on Applied Computing*, pages 893–894, Mar 2009.

[17] A. Deever, and A. Gallagher. Semi-automatic assembly of real cross-cut shredded documents. *IEEE International Conference on Image Processing*, Oct 2012.

[18] S. Shang, H.T. Sencar, N. Memon, and X. Kong. A semi-automatic deshredding method based on curve matching. *IEEE International Conference on Image Processing*, pages 5537–5541, Oct 2014.

[19] K. Shanmugasundaram, and N. Memon. Automatic reassembly of document fragments via context based statistical models. In *IEEE 2003: Proceedings of the 19th Annual Computer Security Applications Conference*, Dec 2003.

[20] B. Gunturk, and M. Feldman. Frequency division multiplexed imaging. *Proceedings of the SPIE Electronic Imaging*, Feb 2013.

[21] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85-104, R. Miller and J. Thatcher (eds.), Plenum Press, New York, 1972.

[22] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, 568 (1964)

[23] B. Golden. A statistical approach to the TSP. *Networks* 7, 209-225 (1977).

[24] D. Rosenkrantz, R. Stearn, and P. Lewis. Approximate algorithms for the traveling salesperson problem. In *Proceedings of the 15th Annual IEEE Symposium of Switching and Automata Theory*, pp. 33-42, 1974.

[25] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. *Management Sciences Research Report* No. 388, Carnegie-Mellon University, February 1976.

[26] C. Kim. A minimal spanning tree and approximate tours for a traveling salesman. *Comp. Sci. Tech. Report*, University of Maryland, 1975.

[27] S. Lin. Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.* 44, 2245-2269 (1965)

[28] B. Golden, L. Bodin, T. Doyle, and W. Stewart J.R. Approximate traveling salesman algorithms. *Operations Research*, 28, 694-711 (1980)

[29] G, B, Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large scale traveling salesman problem. *Operations Research*, 2_t, 393-410 (1954).

[30] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Operations Research*, 11, 972-989 (1963).

[31] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6, 293-326 (1975).

[32] M. Sagiroglu and A. Ercil. A texture based matching approach for automated assembly of puzzles. *Proceedings of 2006 ICPR*, 3:1036–1041, 2006.

[33] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004.

[34] L. Gatys, A. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2414-2423, 2016.

[35] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[36] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, 1991.

[37] D. B. Cooper, A. Willis, S. Andrews, J. Baker, Y. Cao, D. Han, K. Kang, W. Kong, F. F. Leymarie, X. Orriols, S. Velipasalar, E. L. Vote, M. S. Joukowsky, B. B. Kimia, D. H. Laidlaw, and D. Mumford. Assembling virtual pots from 3D measurements of their fragments. In *Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage*, pages 241–254. ACM, 2001.

[38] B. Horn, H. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal for Optical Society of America*, 5(7):1127–1135, 1988.

[39] Q.-X. Huang, S. Flory, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. In *ACM SIGGRAPH*, pages 569–578, 2006.

[40] J. C. McBride and B. Kimia. Archaeological fragment reconstruction using curve matching. In *Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 3–3, 2003.

[41] S. O'Hara and B. A. Draper. Introduction to the bag of features paradigm for image classification and retrieval. *arXiv preprint arXiv:1101.3354*, 2011.

[42] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972.

[43] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. *In Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[44] G. Ucoluk and I. H. Toroslu. Automatic reconstruction of broken 3D surface objects. *Computers & Graphics*, 23(4):573–582, 1999.

[45] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large-scale partial-duplicate web image search. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 25–32, 2009.

[46] Z. Yin, L. Wei, M. Manhein, and X. Li. An automatic assembly and completion framework for fragmented skulls. In *International Conference on Computer Vision (ICCV)*, pages 2532–2539, 2011.

[47] K. Zhang, W. Yu, M. Manhein, W. Waggenspack, and X. Li. 3D fragment reassembly using integrated template guidance and fracture-region matching. In *International Conference on Computer Vision (ICCV)*, pages 2138–2146, 2015.

[48] E. D. Demaine and M. L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: connections and complexity. *Graphs and Combinatorics*, 23:195–208, 2007.

[49] T. R. Nielsen, P. Drewsen, and K. Hansen. Solving jigsaw puzzles using image features. *Pattern Recognition Letters*, 29(14):1924–1933, 2008.

[50] C. Papaodysseus, T. Panagopoulos, M. Exarhos, C. Triantafillou, D. Fragoulis, and C. Doumas. Contour-shape based reconstruction of fragmented 1600BC wall paintings. *IEEE Transactions on Signal Processing*, 50(6):1277–1288, 2002.

[51] B. Burdea and H. J. Wolfson. Solving jigsaw puzzles by a robot. *IEEE Transactions on Robotics and Automation*, 5(6):752–764, 1989.

[52] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan, Solving jigsaw puzzles by computer. *Annual Operations. Research*, 12:51–64, 1988.

[53] J. L. Schonberger and J. M. Frahm. Structure-from-motion revisited. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[54] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal for Computer Vision*, 80(2):189–210, 2008.

[55] A. Willis and D. Cooper. Estimating a-priori unknown 3D axially symmetric surfaces from noisy measurements of their fragments. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 334–341, 2006.

[56] A.R. Willis and D.B. Cooper. Bayesian assembly of 3D axially symmetric shapes from fragments. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:82–89, 2004.

[57] G. Papaioannou, E.-A. Karabassi, and T. Theoharis. Virtual archaeologist: assembling the past. *IEEE Computer Graphics*, 21:53–59, 2001.

[58] G. Papaioannou and E. Aggeliki Karabassi. On the automatic assemblage of arbitrary broken solid artefacts. *Image Visual Computations*, 21: 401–412, 2003.

[59] W. Yu, M. Li, and X. Li. Fragmented skull modeling using heat kernels. *Graphical Models*, 74:140–151, 2012.

[60] X. Li, Z. Yin, L. Wei, S. Wan, W. Yu, and M. Li. Symmetry and template guided completion of damaged skulls. *Computational Graphics*, 35:885–893, 2011.

[61] L. Wei, W. Yu, M. Li, and X. Li. Skull assembly and completion using template-based surface matching. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 413–420, 2011.

[62] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: a general framework for graph optimization. In: *IEEE ICRA*, pages 3607–3613, 2011.

[63] R. Smith. An overview of the Tesseract OCR engine. *IEEE Ninth International Conference on Document Analysis and Recognition (ICDAR)*, 2007

[64] P. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH*, pages 369–378, 1997.

[65] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *IEEE Sixth International Conference on Computer Vision*, 1998.

[66] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. In *ACM SIGGRAPH*, pages 664–672, 2004.

[67] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, pages 71–86, 1991.

[68] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 711–720, 1997.

[69] H. Xu, W. Yu, S. Gu, and X. Li. Biharmonic volumetric mapping using fundamental solutions. *IEEE Transactions on Visualization and Computer Graphics,* 19(5): 787-798, 2013.

[70] X. Li, X. Gu, and H. Qin. Surface mapping using consistent pants decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 15(4): 558-571, 2009.

[71] X. Li, Y. Bao, X. Guo, M. Jin, X. Gu, and H. Qin. Globally optimal surface mapping for surfaces with arbitrary topology. *IEEE Transactions on Visualization and Computer Graphics*, 14(4): 805-819, 2008.

[72] H. Xu and X. Li. A symmetric 4D registration algorithm for respiratory motion modeling. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 149-156, 2013.

[73] X. Li and S. S. Iyengar. On computing mapping of 3D objects: A survey. *ACM Computing Surveys (CSUR)*, 47(2): 34, 2015.

[74] L. G. Brown. A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24(4): 325-376, 1992.

[75] S. Battiato, A. Castorina, and M. Mancuso. High dynamic range imaging for digital still camera: an overview. *Journal of Electronic Imaging*, 459-470, 2003.

[76] Y. Bandoh, G. Qiu, M. Okuda, S. Daly, T. Aach, and O. C. Au. Recent advances in high dynamic range imaging technology. In *17th IEEE International Conference on Image Processing (ICIP)*, pages 3125-3128, 2010.

[77] D. E. Jacobs, J. Baek, and M. Levoy. Focal stack compositing for depth of field control. *Stanford Computer Graphics Laboratory Technical Report*, 2012.

[78] E. E. Fenimore, and T. M. Cannon. Coded aperture imaging with uniformly redundant arrays. *Applied optics*, 337-347, 1978.

[79] J. Chahl and M. Srinivasan, Reflective surfaces for panoramic imaging, *Appl. Opt.*, 8275-8285, 1997.

[80] Y. Ji, J. Ye, and J. Yu. Depth Reconstruction from the Defocus Effect of an XSlit Camera. In *Computational Optical Sensing and Imaging*, pp. CM4E-3. Optical Society of America, 2015.

[81] J. Ye, Y. Ji, W. Yang, and J. Yu. Depth-of-field and coded aperture imaging on xslit lens. In *European Conference on Computer Vision*, pages 753-766. Springer, Cham, 2014.

VITA

Houman Kamran was born in Kerman province of Iran in 1983. He stayed in his hometown throughout his primary education period. After graduating from high school he moved to the capital city of Iran, Tehran, to pursue his studies in computer engineering in Sharif University of Technology. By graduating with a Bachelor's degree from Sharif University, a new chapter begins in his life. He travels across the sea to Baton Rouge, Louisiana, to earn a Master's degree in computer science from Southern University and A&M College and then chase a doctorate degree in computer engineering in Louisiana State University and A&M College.